(2)

USING ELECTRONIC SPREADSHEETS TO SOLVE ENGINEERING PROBLEMS WITH AN INTRODUCTION TO EXPERT SYSTEM TECHNIQUES

by
ROBERT D. GILMORE

A REPORT PRESENTED TO THE GRADUATE COMMITTEE
OF THE DEPARTMENT OF CIVIL ENGINEERING IN
PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF MASTER OF ENGINEERING

UNIVERSITY OF FLORIDA

FALL 1988

ACKNOWLEDGEMENTS

I would like to thank Dr. James Heaney of the University of Florida Department of Environmental Engineering for introducing me to the capabilities of electronic spreadsheets, for allowing me access to a spreadsheet based prototype decision and support system and for co-chairing my report committee. Without this assistance, this report would not have been possible.

Also special thanks in extended to the other members of my report committee: Dr. James Schaub, co-chairman, Prof. Willard Shafer.

A personal thanks is extent to the two graduate students, Jim Wittig and Tom Potter, who are working on developing the studied prototype system. They always took the time to answer my many questions.

TABLE OF CONTENTS

ABSTRACT

Computer spreadsheet software is one of the most important applications for the micro computer. Spreadsheets can be used to conduct a vast majority of engineering analyses in a fraction of the time it formerly took with better accuracy and documentation.

The report has the immediate purpose of describing the general concepts of electronic spreadsheet usage in engineering. In addition, the new concepts of expert system techniques are introduced for the possible incorporation of these techniques into the design of electronic spreadsheet's programming.

The report describes the electronic spreadsheet and presents examples of its usages. The report concludes with a conceptual overview of expert systems. (ES) ←

CHAPTER ONE
BACKGROUND AND INTRODUCTION

Recent advances in micro computer technology have opened up many new possibilities for the application of powerful analytical methodologies in decision making. Small affordable micro computer systems are now available with the computational and data storage capabilities that previously could be found only in the mainframe environment. With these advancements, a wide variety of software applications have come on the market, and commercial computer spreadsheet programs are among the most popular (best selling) applications in this market for the micro computer.

Today, the engineering profession is discovering and using the computational powers of computer spreadsheets in practice. They are used in bid preparation, budget, control, engineering design computation, and many other areas. However, the computational power of the computer spreadsheet is only the beginning of what can be accomplished. The current user friendly menu driven spreadsheet can easily be programmed to help solve many engineering design and management problems, including the checking of computational results and clerical assistance. In the spreadsheet format, the engineer can easily develop stand alone applications without reverting to traditional programming language with its array of input and output statements.

In addition, a new method of computer programming called expert systems may strongly affect the future of computing in engineering. Almost every engineering journal has articles about expert systems. This

unbridled enthusiasm for a new style of computer programming will certainly change the way engineers use computers in the future, and as we will see, the way engineers will be able to use the computer spreadsheets.

In today's environment of stand alone micro computers, and a "do it yourself" attitude towards programming, it is a logical step to consider the insertion of the new computer programming techniques of expert systems into the already proven capabilities of the electronic spreadsheet. This manipulation of current spreadsheet design offers considerable rewards; however, any computer application development or implementation is costly, both in terms of money and engineering time. Therefore, before any undertaking, knowledge about the technique's strengths and limitations should be investigated. The development of a small-scale user designed expert system in the format of the electronic spreadsheet could be the first logical step to learning this new programming technique.

This report will initially explore the use of the electronic spreadsheet. Then, the concepts and techniques associated with expert systems will be looked at.

This report is not a tutorial or reference manual for spreadsheet design nor expert system programing. It was written to provide a simple insight into a new computer technique which offers the engineer a chance to incorporate new ideas into the easily conceptualized domain of computer spreadsheets. The intended audience is the engineer who like the author is initially unfamiliar with the engineering capabilities of the electronic spreadsheet, and who, like the author, is interested in the basic

concepts of expert systems. For the reader already familiar with spreadsheets and expert systems, this report provides insight into what the author, an individual engineer who started this investigation with no micro computer experience, considers to be the important aspects of the subject.

This report reflects the knowledge gained by the author, over the 1988 summer semester at the University of Florida, from working through the Lotus 1 2 3, @BASE, and IF/THEN tutorials, from reading the literature listed in the bibliography, from working with an electronic spreadsheet based prototype decision support system, and from the author's own work with electronic spreadsheets since this time.

1.1 Outline

A micro computer is only as good as the software it utilizes. To understand the basic concept and structure, including expert system programing techniques within a computer spreadsheet, it is necessary to examine the background of computer spreadsheets and expert systems.

Chapter Two provides the introduction to the computer spreadsheet. In Chapter Three, examples of electronic spreadsheet's uses and techniques are presented. This is not only done to reinforce the concepts presented in Chapter Two but to help demonstrate the ease to which spreadsheet application can be applied. In addition, the chapter concludes with a quick overview of an innovative electronic spreadsheet prototype decision support system. This prototype system represents the state of the art in engineering usage of the electronic spreadsheet. For readers

who have further interest in the prototype system, a condensed summary of the system is provided in Appendix II. While the prototype system has not yet incorporated expert system techniques, the system does provide the catalyst for this report. It was the author's hands on experience gained in the investigating of this prototype system which allowed the author insight into the advantages and limitation of spreadsheets and expert systems on the micro computer. Chapter Four will look directly at the techniques and concepts of an expert system. Conclusions are provided in Chapter Five. This report's outline approach was selected to follow the basic learning curve which would be followed by an engineer with no computer spreadsheet or expert system background.

1.2 Two Previous Reports on the Subject of Electronic Spreadsheets and Expert Systems

One previous report presented to the University of Florida's Department of Civil Engineering titled "The Application of Computer Spreadsheets to Stormwater Management in the State of Florida" (Tootle, 1987) concluded that:

-Computer spreadsheets are very useful in design engineering.

-Spreadsheets allow the designer to apply the "what if" principle.

- The use of the computer spreadsheets should expand into all areas of design engineering.

-This expansion would facilitate and improve all phases of design engineering.

The development of this report was on the basics of spreadsheet

programming, and the pros and cons of using such programs (templates) to perform engineering design in the area of storm water management. The primary application is to surface hydrology and the runoff water quality and quantity. The report included seven templates for storm water engineering design. The report stated the main reason for the development of the storm water template is to provide a practical means of designing storm water systems and an output that is clear and easy to understand. The work reported on in this report (Tootle, 1987) reflects the initial or early stages of the prototype system that is commented on in Chapter Three and summarized in Appendix II.

The second report presented to the University of Florida's Department of Civil Engineering titled " Expert Systems in Civil Engineering" (Wall, 1986) concluded that:

-Expert systems hold the potential to herald a revolution greater than that introduced by the micro computer.

-In building an expert system tailored to a particular environment the price could easily run in excess of a few hundred thousand dollars.

-Expert systems are costly in terms of money and time; therefore, it is important that expert systems are developed and implemented only within those fields (domains) where their strengths can be exploited, and their cost can be justified.

- Expert systems will allow the true capability and potential of micro computers to be utilized; for the first time, there will be application programs available that actually assist the user, and do not simply regurgitate the input data in a disguised form.

-And that due to this as well as the requirement to assemble and maintain a staff of experts during the development, that most companies will not implement expert system until stand alone, off the shelf programs become available at a reasonable price.

The development of this report was on the basic concepts and structure of expert systems. It examines the fundamental theories of artificial intelligence, and the utilization of these techniques as the nominal component parts of an expert system. The report did a case study of two existing prototype expert systems.

The conclusion of this report (Wall, 1986) lends support to the idea of incorporating the expert system in spreadsheet programming as an inexpensive and an easy way to start taking advantage of these new programming ideas.

1.3 Subject Overview

1.3.1 Micro Computer

Large main frame computers have been available for many years. However, the expanding technology in the electronics field has brought about a revolution of small size, relatively low cost computers and other hardware that has created today's environment of stand alone desk top micro computer systems. Micro computers provide work stations at which activity can be performed without the need for the extensive use of the traditional pencil, paper, and hand held calculator.

An additional development fueling the increasing use of the micro

computer is the user friendly atmosphere of today's software applications. For years, the computer rooms in the large companies were manned by dedicated experts who were protected by their unique jargon and surrounded by the mystique of the computer. Now, even grade school children are taught the fundamentals of computer use and programming. The jargon has become part of everyday language, the mystique has been replaced with a "do it yourself" attitude, and a virtually unlimited amount of low cost, off-the -shelf user friendly software is now available (Gifford, 1987).

As hardware technology continues to advance and software writers utilize these advances (increased operating memory, faster processing time, greater data storage, etc.) in their programs, the applications of the micro computer will be limited only by what can be imagined. However, attempting to enumerate all the uses of a micro computer is futile.

1.3.2 Computer Spreadsheet

The computer spreadsheet program was originally created as a financial management tool which provided a means of examining numerical data on a mirco computer. However, spreadsheet programs have become some of the most popular pieces of software for the micro computer due to their general applicability to solve a wide range of problems. The results are easy to visualize, and the application is easy to learn and use.

The Lotus 1 2 3 spreadsheet is the most popular spreadsheet and is universally available. Lotus will do analysis and forecasting of numerical data, data base management, and data presentation (graphics), thus the

term 1 2 3. With fast computational speed and clearly tabulated and plotted results, the Lotus spreadsheet has proven to be a valuable tool for engineers. An introduction to spreadsheet analysis and Lotus 1 2 3 is presented in Chapter Two.

1.3.3 Expert System

Expert system technology is a new class of computer program which comes from a branch of computer science that is referred to as Artificial Intelligence (AI). AI is concerned with a broad range of topics that are related to simulating human intelligence in a computing machine. Some of the better known areas of AI are natural language understanding, machine vision, robotics, and expert systems. Expert systems are a result of many attempts over the years to simulate or reproduce intelligent problem solving behavior into a computer program. They use symbolic knowledge to simulate the behavior of the human expert with the idea being that the computer will be able to interact with the user in much the same way that a human consultant does. However, current expert systems are simply nothing more than new applications of computer programing techniques which do not meet the above descriptions (Partridge, 1986).

An introduction to expert system's concepts, basic components, and methods is presented in Chapter Four.

CHAPTER TWO
COMPUTER SPREADSHEETS

The primary objective of this report is to investigate the abilities of the electronic spreadsheet software in the solving of engineering problems. Also, the ability of the spreadsheet to use the expert system technique of symbolic computation for problem solving is investigated. Therefore, this chapter reviews the electronic spreadsheet software Lotus 1 2 3, with an emphasis on the symbolic computation, advance programming language (Macros), and after market add on, all of which, allows Lotus to use some of the techniques of expert systems. It is not the report's objective to teach basic spreadsheet programming; however, a basic understanding of the computer spreadsheet is required and is presented.

2.1 Introduction to Lotus 1 2 3

The computer spreadsheet is a computer program (software) designed for analyzing and forecasting of data. It provides a basis for information management and presentation. While there are more than eighty spreadsheet programs available on the market, this report is confined to an investigation of Lotus 1 2 3, the most popular and widely used spreadsheet program. Lotus has proven to be an excellent program for engineering design and problem solving. It has built-in statistical, scientific, financial, and graphical commands which facilitate solving engineering problems. It can reduce the time required for proposal writing, bid preparation, expense estimating, data presentation, and for calculation and design optimization. A powerful application is its ability

to quickly make revisions to, or play "what if" with any problem and quickly show the

results in a usable format. This makes sensitivity analysis for any input value extremely easy, because only that one value has to be changed for Lotus to recalculate the spreadsheet. Additionally, Lotus has data table capabilities in which an entire range of input values can be entered at one time, and Lotus will calculate and display all the results in table format. However, the major advantage of the electronic spreadsheet is that it is so easy to use and to visualize the results.

2.2 What is a Computer Spreadsheet?

A computer spreadsheet is the same as a accountant's spreadsheet. It is a matrix (grid) of columns and rows of "cells". This accountant's electronic ledger (Lotus) has 256 columns and 8,192 rows or 2,097,152 cells. Each cell can hold a value of a number, a label (text), or a formula. The matrix is labeled across the top (columns) by letters and down the side (rows) by numbers. Thus, using one letter followed by one number will address or refer to one cell. In addition to the cell's column and row address, each cell can have a name or attribute assigned to it (range name).

An analogy to the computer spreadsheet would be a very large piece of grid paper equivalent to about 12000 sheets of conventional paper, in which a wide variety of numbers, formulas, or text can be entered into each of the grids' cell simply by moving to the cell and typing in the data. The cells can be expanded by column width to display entries, moved (cut and pasted), or copied into another part of the grid paper.

From the above analogy, it is apparent that using a spreadsheet program has numerous advantages: data entry is easier, computations are fast and accurate, and the presentation of results is cleaner. Another significant advantage is that the data is only entered once. Separate input/output statements are not needed in the spreadsheets, i.e., no "Black Box" effect occurs. The output is visible on the computer's monitor. To view what input produced an output, the underlying contents (formula) of the output cell can be viewed. However, if macros are used, the black box effect returns and the input associated with an output may not be obvious.

Every cell has two parts : 1) the cell's address or name, and 2) the cell's contents or value. Spreadsheets do not display formulas (only their values), unless instructed to do so. An example would if the cell A5 containing the formula + A3 +A4 where the value of cell A3 = 8 and the value of cell A4 = 7. The value shown on the screen would be: the output value 15 for cell A5. However, to see the formula (+A3 +A4) the cursor (pointer) would have to be moved to cell A5 to see that its content is a formula.

A "template" is a Lotus program file that contains a program previously input to the spreadsheet. The template contains values, formulas and labels that solve a specific problem. It is stored under a specific Lotus file name, and is retrieved by that file name. A template is used the same as any other computer program, except the results and the program are just one spreadsheet. The resulting spreadsheet can then be saved or discarded as any other file while the original template file remains saved and ready for further use. On many occasions, the term spreadsheet and template are used interchangeably. Remember, a template

is a file developed to perform a specific task.

The following three sections provide specific information about Lotus's basic concepts, macros, and add ins (@Base).

2.3 Basic Lotus 1 2 3 Capabilities and Symbolic Representation

The purpose of this section is to provide a general familiarization of Lotus's basic capabilities. The interested reader should consult the Lotus 1 2 3 reference manual or one of the many good after market books like "Master 1 2 3" by Jorgensen (1988). Lotus is very simple to use and to program. However, it is helpful to have some previous experience in computer programming. Lotus uses the same computer logic and hierarchy of operation that are found in most computer languages.

The reader is or can easily become familiar with Lotus's menu commands, cursor movements, and the Lotus function keys (F-Keys). A complete on-line help index and manual is contained within the Lotus software. This Help reference is available by pressing the F-1 key at any time. Lotus has many built in applications (@ Functions) that facilitate the use of the program. Many steps can be saved by using these built in functions along with increasing the speed by which the program will run.

A common use of Lotus is to enter (type) numbers, text, Lotus @functions, or formulas directly into a cell. Formulas can produce labels (words) or values (numbers).

Variables entered into a spreadsheet are identified by cell address or can be "named". The range name command is used to name a cell or a range of cells. This important feature of Lotus allows the formula relationships of name cells to stay intact if cells are moved. Also, this naming

capability is very important because expressions (formulas) that are reasonably understandable using named variables (cells) are incomprehensible when cell addresses are used. One of the initial goals of expert system programing is constructing programs in which the expressions have the desired meaning in human language as well as in computer program language (symbolic representation). The ability of Lotus to represent words, ideas, concepts, and relationships, and to compute using them is the key feature which allows the programming of expert system techniques which rely on symbolic representations and relations. Labels (text in a cell) are a simplified example of this idea (If/Then Solutions, 1987).

More complex examples are formulas of string arithmetic where Lotus will "string" labels together to create a new label in a new cell. Values can also be combined with a string of labels (versus numerical formulas), if the value (number) is first converted to a label using the Lotus @ String function.

String formula starts like any Lotus formula with a plus sign (+). Each string that is typed explicitly is surrounded by quotations and the character "&" is used to link each string. The standard form for string addition is:

+ string 1 & string 2 & string 3 &... & string n

where a string can be:

1. Any label defined by quotes.
2. A formula resulting in a label.
3. Any @ function that results in a label.

String (label) must not be confused with a cell name (range name). The

Label represents the value of the cell, while the range name represents the attribute (address).

2.4 Lotus 1 2 3 Macros

Lotus macros consist of a series of instructions recognized by Lotus and entered as labels in a vertical series of continuous cells. Macros are literally keystrokes that (with more difficulty) could have been made by the person at the keyboard. They are simply mini-programs or individual subroutine programs built within the Lotus spreadsheet. Macros are given the range names (using Lotus's range name command) of a "backslash" and a single letter. Macros are started by pressing the ALT key and the macros range name (single letter). This transfers control of the spreadsheet to the marco's commands which are executed in series. Lotus macros may either be Lotus menu commands or may be Lotus Advance Command Language (ACL) commands.

This programming language of macro instruction (ACL) allows the spreadsheet to become comparable to traditional programming languages. Macros written with ACL can retrieve files, present selected cells for data entry, prompt the user for information in straight forward English, control the program's flow and functions, manipulate data, and controls the computer screen. Because macro ACL commands are numerous, a complete discussion is not presented, but examples of macro the author created are presented in the next chapter.

The two ACL macro commands which are important to the expression of rules in expert systems are the IF command and the LET command macros. The syntax is indicated below:

{IF condition}

{LET location, number or string}

The IF macro checks the stated condition for being logically true or false. If true, then the next macro command is executed. If false, then the next macro command is skipped. The LET macro places a number (value) or a string (Label) in the stated location. The IF macro is exactly analogous to the if in the if-then rule, and LET macro in this context could be rephrased then-let (IF/THEN Solutions, 1987).

While the Lotus spreadsheet would still be a powerful tool with out macros, it is the macro which truly represents the state of the art programing ability of Lotus.

2.5 Lotus After Market Additions

No program can do everything. However, Lotus is fortunate to be the leading and dominate spreadsheet in use today. For this reason a host of other software manufactures have designed products to enhance Lotus (Jorgensen, 1988).

These compatible software products are categorized as, Lotus "Add-On" and "Add-In". Add-Ons' are stand-alone programs which easily exchange data with Lotus spreadsheets. Add-Ins' are programs which operate from within Lotus. They can be RAM-resident programs which share memory and "pop up" when called for, or they can be preprogrammed templates which contain macros, complex formulas, and/or instructions (expert system shells). These products enhance Lotus by providing powerfully integrated specialized applications which include such items as: word processing , printing, data management, desktop management,

networking, data measure collection, customized report formatting, debugging and documentation tools, drawing/graphics packages, charting and graphing upgrades and expert system shells (Jorgensen, 1988).

Additionally, a program called HAL adds a variety of time saving features by incorporating some artificial intelligence programming techniques. With HAL a multitude of symbolic command (language) can be used to instruct the computer. Like other programming languages HAL has its own special language. But its language consists of every day English verbs. This ability allows instructions to be issued to the Lotus spreadsheet in a more natural format. While this type of instruction requires more key strokes, it has the advantage of presenting instruction in an easily comprehensible format, and as will be seen, this is an important feature in expert system development. While this report does not investigate the use of HAL, HAL appears to offer features which should be considered desirable in Lotus spreadsheet programming (Jorgensen, 1988).

While new programs are coming on the market almost daily that will continue to expand the range of application for Lotus, an add-in program called "@Base" is what really allows the practical advancement of Lotus program to handle large amounts of data. @Base allows data base storage to be kept on hard storage and accessed when required versus the Lotus's built-in data base which requires that the entire data base be present in RAM (stored in the spreadsheet). The outcome is that @Base allows Lotus to create and use large data bases not limited by the micro's RAM limitations but only by the virtually unlimited data capacity of the micros system and by the time limitation of the current application.

"@BASE is a powerful database manager designed especially for Lotus users. @BASE compares with popular stand-alone database programs like dBASE, but operates inside Lotus" (Personics Corporation, p.1, 1987). @Base is a database application which adds into Lotus and allows browsing, storage, query, data transfer to and from (import and export) the spreadsheet and disk storage. Lotus-like menus make @BASE easy to learn and use. Data capacity stored on disk can grow as large as 32 megabytes. Also, with @BASE you can access multiple data files at the same time which is beyond the current capabilities of Lotus 1 2 3, release 2 (Personics Corporation, 1987).

Disk-based storage does more than expand data capacity. It separates logic of the database model from the data, making the model easier to understand and more viable for expert system implementation as this is exactly the method used in expert system designs. The knowledge of the problem is separate from the solution method. In addition, @Base provides index modules which provides faster data retrieval by allowing files to be organized and searched by one or more "key" fields called "indexes". Once a data file has been indexed, @BASE can build a separate index data file that contains only the key field for each record along with a pointer that tells the program where the complete record is located. Because of their smaller size and organization, index files can be searched more quickly than normal data base files. This is an important feature for any large relational data base system (such as those required in an expert system). In general, @BASE doesn't reproduce any function which can already be performed by Lotus. Lotus macros can be used to drive @BASE applications and data bases (Personics Corporation, 1987).

2.6 Summary

Lotus is a powerful computer software application widely used in industry. Lotus is an electronic spreadsheet whose maximum size of 8192 rows and 256 columns does not appear to pose any space limitations in spreadsheet design (templates). However, the hardware limitation on spreadsheet size depends upon the micro computer's memory (RAM) limitations. Lotus's basic operations, logic, and hierarchy are very easy to learn and use.

The idea of symbolic representation where individual cells can be addressed by name, and where labels of different cells can be strung together is a key concept in an expert system development.

Lotus contains a very powerful programming language which is used in the creation of Lotus macros (programs). Macros allow control of the spreadsheet to be turned over to the computer until the macro has completed its program.

There are many after market additions to Lotus. The add-in @Base which allows disk storage, multiple opening and closing of data base and the indexing of related data bases is a major component in the ability to create large integrated Lotus applications on a micro computer as will be shown in Chapter Three. In addition, Chapter Three provides examples and discussion on the author's own attempt to use the information presented in this chapter. Chapter Four will demonstrate the basic concepts of expert systems and how Lotus's symbolic representation, macros, and add-in @Base can be used to implement these new programming technique called expert systems into the spreadsheet.

CHAPTER THREE
THE PROCESS OF INTEGRATING EXPERT SYSTEMS TECHNIQUES INTO A SPREADSHEET

While the majority of this report's contents are basically a generic overview of electronic spreadsheets for the solving of engineering problems with an introduction to expert system techniques, this chapter represents a study of the author's involvement with the subject. This chapter should help the reader understand and appreciate the powerful potential of current computer spreadsheet technology and to help the reader view the practical application of these concepts by demonstrating some of the concepts explained in Chapter Two. Also, this chapter provides a background for the explanation of expert systems to be covered in Chapter Four.

The catalyst for the author's involvement with the engineering usages of the spreadsheet has been the opportunity to observe the development of a project for a decision support system for reviewing surface water permit applications.

This project is the current state of the art in the engineering use of computer spreadsheets. This prototype system is part of a project funded by the South Florida Water Management District is being developed under Dr. James Heaney, Professor, Environmental Engineering Department of the University of Florida.

3.1 Introduction

The author's initial goal in observing the prototype system was to create an expert system checking component. To achieve this goal, the

author followed the same format or learning process presented in this report and identified the following sub-goals: 1) learn the basis of electronic spreadsheets, 2) learn to use and write macros, 3) learn the Lotus add-in @BASE, 4) learn to work with the prototype system, 5) learn expert systems techniques, and 6) integrate expert system techniques into the prototype system. While the author did not achieve the stated goal in the short time frame available, it was through the process of developing this goal that the author went from knowing very little about or nothing about micro computers, electronic spreadsheets, and expert systems to the basic understanding of the knowledge presented in this report.

The remaining sections of this chapter discuss, 1) examples of the author's own uses of basic engineering programming of electronic spreadsheets to illustrate sub-goal number one, 2) examples of simple macro construction to demonstrate sub-goal two, and 3) discussion of the author's work in macro development for the prototype system to show the success of sub-goals three and four. The sub-goal number five is presented in Chapter Four. The final sub-goal number six was not accomplished and is only hinted at in the discussion in Chapter Four. A summarized discussion of the prototype system is presented in Appendix II.

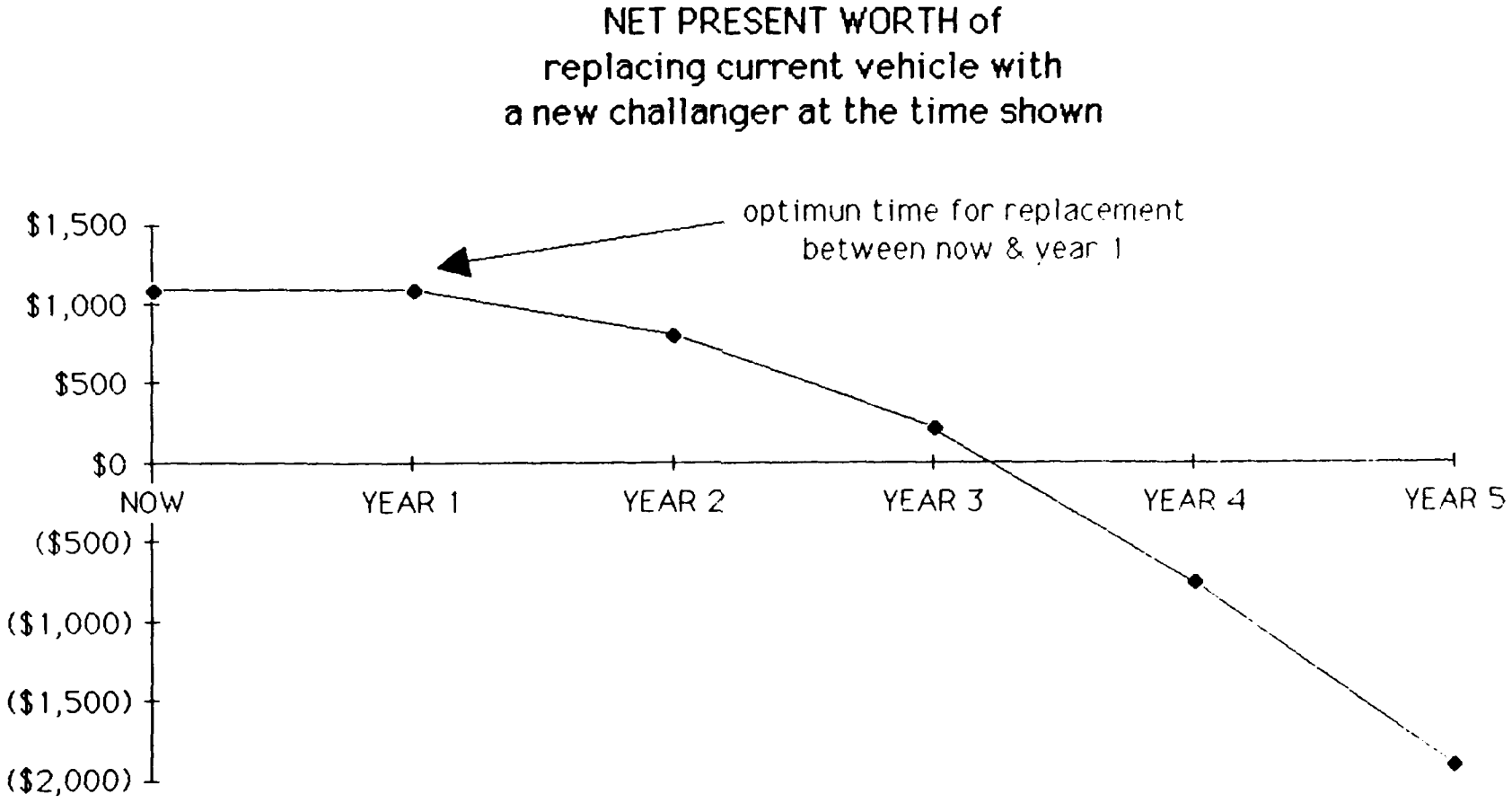
3.2 Basic Spreadsheet Engineering Programming

This section of the report is used to demonstrate the capabilities of electronic spreadsheets to solve engineering problems and the ease at which electronic spreadsheet concepts can be learned and applied. This example is representative of the applications the author has applied the

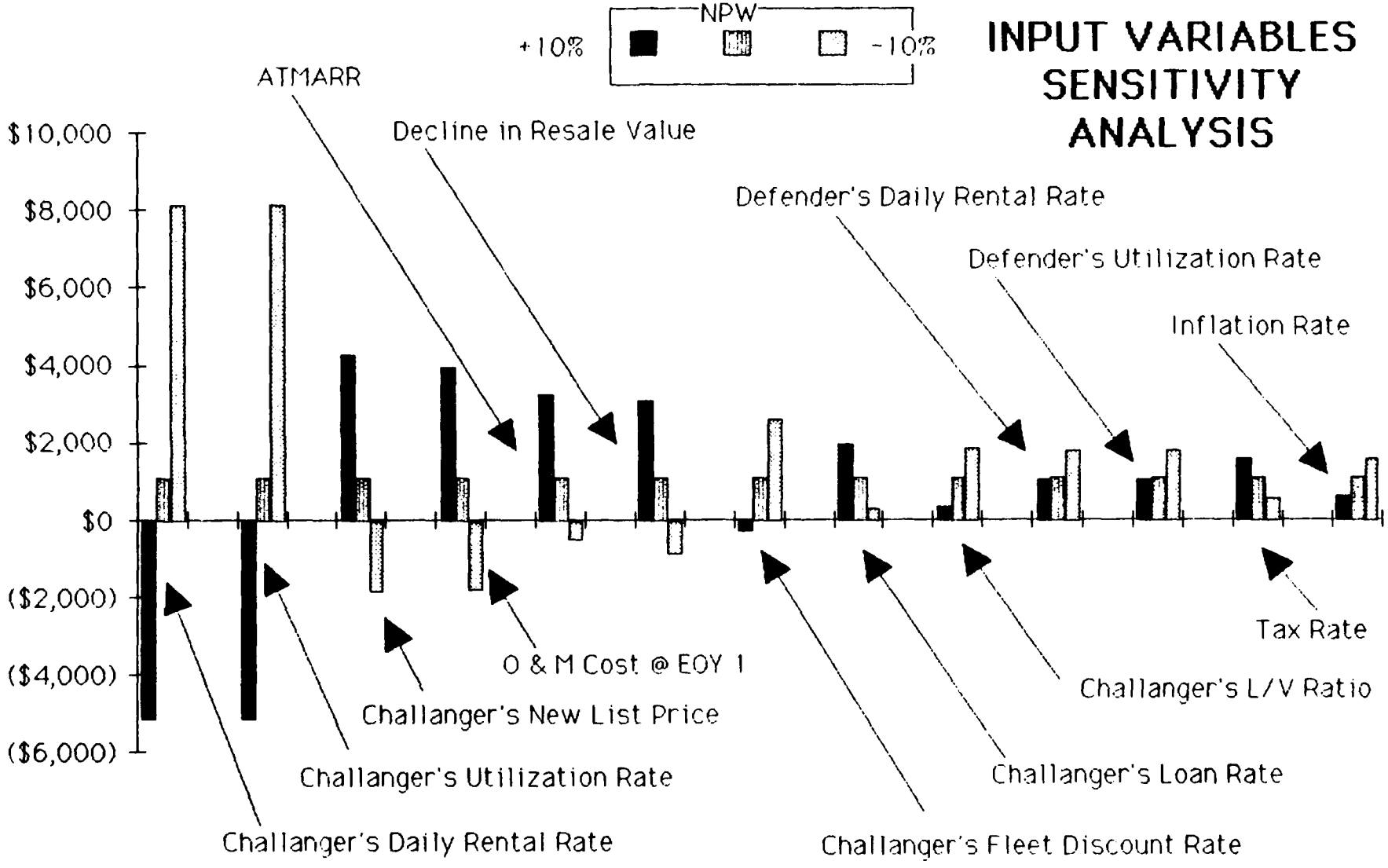
electronic spreadsheet to since being indoctrinated to its usage. The example is the author's refinement and additions to a spreadsheet program taken from the second edition of "Engineering Economic and Cost Analysis" by C. A. Collier and W. B. Ledbetter. It was written for a class project which required the optimum life of an existing piece of equipment to be determined based on the replacement's cost and life cycle. A complete print-out of the spreadsheet is contained in Appendix I. A review of the print-out in the appendix would show the ease of spreadsheet documentation of a program. The input cells are clearly defined as are complete explanation of the program's equation. The program is an economic feasibility program which uses the built in statistical function of the spreadsheet to create a matrix of net present worth for varying times to replace current equipment (defender) against varying life cycle time of replacement (challenger). The highest number in the matrix represents the optimum time to replace the defender with a challenger.

With this spreadsheet program, the engineer can easily evaluate equipment replacement, determine after taxes rate of return on the proposed investment of new equipment or keeping the existing, determine present value of the proposed cash flow, determine the cost penalty for keeping or trading in early the existing equipment from the optimum time, determine the appropriate cost for the equipment's rental rate, and determine the sensitivity of the input values. All of the above are readily available from a spreadsheet template which took only a day to design.

This program clearly demonstrates the electronic spreadsheet to be a powerful analyzing tool for engineering work. Graph 3.1 and 3.2 are good examples of the graphical representation available from electronic



GRAPH 3.1: Sample Electronic Spreadsheet Graph



GRAPH 3.2: Sample Electronic Spreadsheet Graph

spreadsheets. Graph 3.1 shows the results of the present value of keeping or replacing the defenders for the example problem in Appendix I. Graph 3.2 shows the sensitivity of the program's input values.

3.3 Lotus Macro Development

This section explains simple Lotus Macro construction and shows examples of these macros. Figure 3.3 is a print out of simple Lotus macros written to save key strokes during spreadsheet construction. Notice the format used. Each of the macros are numbered and have a descriptive text. Next, the key "Letter" which will activate the macros is displayed. Remember, macros are started by the user pressing the ALT key and the macro's range name (single letter). Finally, the macro commands themselves are displayed as labels.

To create a Lotus macro requires nothing more than to enter Lotus menu commands or programming language into a cell or a series of vertical cells (columns) and give the cell or first cell in the series a Lotus range name of a backslash "\" and a single letter. Of the macros shown in Figure 3.1, macro numbers 1, 2, 3, 4, and 6 are nothing more than saved Lotus menu commands which could manually be entered by the user each time the macro's function is required. However, the macro saves the user time as the following examples will illustrate (see Figure 3.1): Macro number 1, file save, saves the spreadsheet to the system's hard drive (c) and to a back-up "floppy" disk in drive (a) any time the user types ALT "F". The macro command "/" calls up the Lotus menu, "F" selects the Lotus's file option, "S" selects the Lotus's save option, "{ESC} {ESC} {ESC}" clears Lotus's default setting for the location of the spreadsheets file, "A:sat_1"

Library (macros):

	Description	Key	Macro
1.	File Save	\F	/FS {ESC} {ESC} {ESC}
	to A drive		A:set_1~R
	to C drive		/FS{ESC}bob\sfw\set_1~R
2.	Label Range	\L	/RNLR~
3.	Erase	\E	/RE~
4.	Insert Row	\I	/WIR~
5.	Set Column Widths	\W	{GET LABEL "Which Column?", col}
			{GET LABEL "How Wide?", width}
			{GOTO}
		col	ep
			1~/WCS
		width	7
	yes		~{GET LABEL "Press N <Enter> to quit press <Enter> to continue", yes}
			{if yes=" "} {Branch \W}
			/REyes~
			{GOTO} {NAME} {NAME} {?}"
			{QUIT}
6.	Quit 123	\Q	/FS{ESC} {ESC} {ESC}
	saves file to A drive & C drive		A:set_1~R
			/FS{ESC}bob\sfw\set_1~R
			\QY

Figure 3.1:
Simple Lotus Macro written to save key strokes during spreadsheet construction.

says save file named sat_1 in drive (A), "~" represents pressing the keyboards <enter> keys, "R" selects the Lotus's replace option, and "/FS{ESC} bob\ sfw\sat_1 "R" repeats the whole process to save the spreadsheet to the hard drive. The only difference is the "bob\sfw\sat_1" which instructs Lotus to the location of the file sat_1 (ie. sat_1 is located the in directory bob and in sub-directory sfw). From this small macro it is easy to see the huge amount of typing time and delay time between commands which can be saved by the use of a single macro. Also, this particular macro is important in spreadsheet development for it automatically makes a backup copy of the spreadsheet to a floppy disk just in case the system "crashes". Macro number 5, in Figure 3.1, represent a little more complex macro as it uses Lotus's menu commands advance programming language and user's input. This macro sets column widths on the spreadsheet. Again, this macro's functions could manually be accomplished by the user. This small macro is an excellent example of a spreadsheet's ability to interact with user and his input to proceed with a task.

The macros command "{GET LABEL "Which Column?", col}" asks "which column?" and takes the answer and places it in the Lotus cell with the range name (col). "{GET LABEL "How Wide?", width}" asks the question "how wide?" and takes the answer and places it in the Lotus cell with the range name (width). "{GOTO}" represents the F-5 key on the keyboard and is self explanatory. "ap" is the value of the cell with the range name of col which came from the user. "1~" finishes the GOTO command by instructing Lotus to go to the first cell in the column (ap) selected by the user. "/WCS" are Lotus menu commands for the options of worksheet, column,

and set width. "7" is the value of the cell, with the range name of width, which came from the user. "~ {GET LABEL" press N <enter> to quit or press <enter> to continue", yes}" completes the set column width command and asks the user if he wants to continue setting column width or to quit the macro and then places the user's answer in the cell (yes). "{if yes =" "} {BRANCH \W}" checks the value of cell (yes) if the value is blank Lotus branches back to the beginning of the macro and starts the macro over, and if the value is anything besides blank the macro skips the remainder of this command and goes to the next cell down for additional instructions. "/RE yes~" erases the contents of the cell (yes). "{GOTO} {NAME} {NAME} {?}~" represents pressing the F-5 key and the F-3 key twice which will present the user with a complete list (library) of the spreadsheets assigned range names, and the ACL {?} is a pause which allows the user to select the range name location desired in the spreadsheet. "{QUIT} instructs the macro to stop (a blank cell has the same effect).

While the detailed explanations of the above macros were tedious, it does provide insight into Lotus' macro development. The last point of this section is that once a library of macros like those in Figure 3.1 have been created, they can be placed out of the way in a corner of the spreadsheet. Also, this library of macros can be transferred to any spreadsheet under development. Only the macro's first cell would have to be re-named in the new spreadsheet, and macro number 2 could do that task.

3.4 Prototype Decision Support System

This section looks at the author's involvement with the prototype decision support system described in Appendix II. Prototyping is an

interactive style of programming between the developer and the end user that requires modification to the program in response to the user's reactions to trial modulars or components of the system. As previously stated, the prototype system was the catalyst for this investigation. In learning the ins and outs of working with a large prototype system and trying to develop an expert system checking component, the author developed several macros. This section will describe these macros; however, the tedious task of working through the macro commands which was done in the last section will not be attempted.

The prototype system is a large interactive computer program which uses and integrates commercially available software developed to run on the micro computer. The primary application software is Lotus 1 2 3 with the add-in @BASE. The secondary software for letters and report generation is the wordprocessing application WordPerfect. The system is designed to act as a stand alone operation for an engineer who is required to review surface water permit applications.

The reviewing engineer has 20 to 30 active permit applications open at any one time. Each stage of the review process has stringent time frame requirements which must be met by the reviewer. The review requires both administrative and technical engineering design calculation.

The goal of the prototype system is to integrate all of the review functions into a single system to reduce the time required by the engineer to review each permit throughout the process. To accomplish this goal, the system was developed by modular or stand alone components. Most of the components consisted of several Lotus templates. The underlying link to the different system components was the development of relational

data bases for the entire system. These data bases are @BASE file controlled and operated from the Lotus spreadsheet. The data bases are linked to the system's components. Therefore, once input data has been entered, it is available for each component without having to re-enter the data. This feature alone represents a significant saving of the reviewer time. In addition, the storing of the component's output in the same data bases provides the ideal opportunity for the data's verification and reasonableness in the checking.

At the time of the author's involvement with the prototype system not all of the components were taking and giving input and output to the data bases, nor was any global Lotus template available to operate the system. Therefore, the author's goal was to create a global template which would act as a checking component for information storage in the data bases. In any data base system, it is the validity of the records in the data base which support the entire system. While some of the individual component's templates had some internal checking capabilities, the data in the @BASE file could easily be edited, modified, or deleted. This further suggests the need for a checking component for the data bases. In addition, it is important that any stand alone system should be able to give the current status of all or any permit application in the systems.

With the above thoughts, the author created several macros which would automatically attach @BASE and open the system's data base files whenever the spreadsheet file is opened. In addition, the spreadsheet was automatically date stamped, password protected, and given custom menu commands. A password protected system was required if the integrity of the system's data is to be maintained. If the user was to leave the custom

menu commands without first giving the password, the system presented the user with an error message and then returns the spreadsheets control back to the custom menu. The last macro created allowed the user to select any of the system's data bases again and to get a listing of the permit application numbers in the data base. From the list, the user was then free to select an application number, and the macro would inport that application number's data record into the spreadsheet. This was considered the first step in the checking process. This is as far as the author's checking component was developed.

While the above prototype system is highly specific and inflexible, it does provide a view of the current engineering use of the computer spreadsheet and does offer insight into the practical uses of expert system ideas as they correspond to conventional programming applications.

An expert system will augment the professional judgement of the reviewing engineer by being able to quickly access and bring to light past decisions (corporate knowledge) contained in the newly created data bases. However, a more important benefit will be the expert system's ability to check and validate the data records in the data base and call the result to the reviewer's attention. While it is easy to conjure up a variety of benefits of an expert system for the surface water permit review process, there is still the question of whether the expert system program can be created to perform and interact with a current spreadsheet template without disruption. The outlook is bright that the future will hold the answer to be yes.

Of course, the above represents only a limited view of the potential

that expert system techniques have to offer. Expert system features are introduced in the next chapter with an emphasis toward the application of spreadsheet programming. However, the major emphasis will be on a generic overview of expert systems.

CHAPTER FOUR
EXPERT SYSTEMS

In the previous chapters an overview of the electronic spreadsheet Lotus and some examples were presented. The attempt of this chapter is to look into the future engineering uses of the micro computer and particularly the future engineering uses of the electronic spreadsheet. To accomplish this goal this chapter will discuss the concepts of expert systems and look at the building and using of expert system strategies in Lotus spreadsheet designs. The concepts and ideas of integrating expert system technique into the spreadsheet are from the IF/THEN user's manual "IF/THEN... The Hands-On Introduction to Artificial Intelligence Using Lotus 1 2 3" (If/Then Solutions, 1987).

The stated goals of the IF/THEN manual are:

"1) Understand many of the key concepts of AI.

2) Become confident of how the concepts may be used.

3) Build simple AI ideas into spreadsheets so they can solve problems previously impossible to solve with conventional spreadsheets" (If/Then Solutions, p.1X, 1987).

An attempt will be made to provide conceptual insight into the above goals of IF/THEN. It is not the intent of this chapter to duplicate the IF/THEN tutorial or reference manual material. Detailed explanation and tutorial examples in the form of Lotus templates can be found in the IF/THEN software package.

The described information contained in this chapter is an accumulation and summary of the knowledge the author gained by reading the listed material in the bibliography. The author has not personally built

nor developed an expert system. This chapter should be considered an introduction to expert systems based on a literature review conducted by the author. The first section of this chapter is an introduction to expert systems. The following sections will discuss expert system knowledge representation, desirable characteristics, architecture, and problem solving control strategies.

4.1 Introduction

Expert systems have received considerable attention raising the hopes and the potential for a successful application. The expert system is a relatively new methodology in computer programming that provides an infusion of expert knowledge to a problem. It is an extension of basic computer programming principles to a new level of sophistication. Expert systems are defined as a system technique that use logical relationships (judgements), to incorporate past knowledge and expertise about a specific problem area to perform specialized tasks that typically require human expert judgement (Maher, 1987).

Expert systems incorporate traditional rules of thumb and knowledge, that are common in the decision making process about a specific subject into computer format which allows a computer program to access the information. This information along with user input is used to reach a conclusion or statement of an idea.

Expert systems differ from other forms of programming in the clear distinction between their problem solving strategy (inference engine) and the knowledge about the problem (knowledge bases). Also, they relax the sequential routine (algorithmic method) of conventional computer

programs, with programming oriented towards numerical processing, with a programming technique that is orientated toward symbolic processing.

Symbolic processing implies that facts, observation, and hypotheses are represented by words and are manipulated as words. The symbols are stored as rules. By transformation of these rules, an expert system is able to convert the user's and spreadsheet's inputs into some conclusion.

Although Lotus is a good language for learning (AI) expert systems methods, the size of program that will run in a reasonable amount of time is limited by the power of Lotus on today's micro computer. Therefore, Lotus facilitates only the building of small systems to solve small specific problems (If/Then Solutions, 1987). The remainder of this chapter will explore the basic concepts of expert systems in enough detail to allow the reader an insight into subject.

4.2 Knowledge Representation

Building an expert system is in a sense creating a model of an expert's thought process. The system must focus on a single area and restrict itself to a single domain of expertise. For a domain (field) to be suitable for expert system representation it must exhibit the following six necessary domain criteria (Wall, 1986):

1) Genuine experts must exist.
2) The experts must generally agree about the choice of an acceptable solution.
3) The experts must be able to articulate and explain their problem solving methodology.

4) The problem of the domain must require cognitive, not physical skills.

5) The tasks cannot be too difficult.

6) The problem should not require common sense or general world knowledge.

These above criteria restrict the implementation of expert systems to those fields where the basic knowledge is fairly stable and unchanging.

While domain characteristics are important for a successful expert system, the high cost of development and implementation make these criteria only necessary conditions and not sufficient unto themselves to justify development (Wall, 1988). However, the development of small scale expert systems using a shell would represent minimal cost compared to the amount of insight and knowledge to be gained.

Given a suitable domain, the knowledge must still be represented by the computer's software in a fashion that allows it to be utilized. Currently three methods are used for representation: 1) systems that represent their knowledge in a rule type format, 2) systems that rely on a SEMANTIC NETWORK to organize their knowledge and, 3) systems that utilize (a combination of the preceding two) frames" (Wall, pp. 20, 1986). Frames and sematic network will not be discussed. Our emphasis will be on the basic ability to derive knowledge from rules.

Rule based systems represent knowledge in an IF-THEN format (i.e. IF [condition true] THEN [action]). This is the simplest and most widely used representation strategy (Wall, 1986). It is also the strategy used in the spreadsheet design of an expert system. It has the benefit of making modification, deletion, or adding of rules relatively easily. Unfortunately,

this can also be a liability as it is very easy to introduce rules which contradict each other.

Lotus can represent words, concepts, and relationships, and to compute using these relationship. This is done by the simplest form of knowledge representation, attribute-value representation. The attribute is the name of cell and the value is the contents of that cell. Examples of attribute-value representation of several English sentences are shown below (If/Then Solutions, p. 2, 1987):

English Sentence	Attribute	Value
Today is Wednesday.	Today	wednesday
The temperature is now 82.	Temperature	82
The car is red.	Color_of_car	red
The sales to date are 4800.	Sales_to_date	4800
David is happy.	David	happy
The weight of part 33 is 302	Wght_of_prt_33	302

In Lotus, the attribute is the range name of the cell, and the value is the data in the cell. With the attribute-value representation in Lotus, there can only be one value for each attribute (one value per cell). Symbolic computation works by comparing these values (strings of characters). In Lotus these strings have to be exactly equal. Lotus cannot check cell values to see if the values are "synonyms". This limitation will allow only simple small scale expert systems to be constructed in Lotus (If/Then Solutions, 1987).

4.3 Desirable Characteristics

First of all, an expert system must perform well on difficult

problems. Mediocre performance would render it unreliable and performance restricted to easy problems would render it useless. The second practical requirement is that the system must be implementable. The reason for this is that at the amount of knowledge of even a narrow domain requires a lot of effort to get hold of, to get into a working state, and to get it right. The third requirement is that the system must converse in terms that the user can understand and in terms relevant to the problem at hand. The last requirement is expert systems must be able to explain themselves: 1), a system must be able to explain how it reached its conclusions from the facts given. If it cannot, there is no way to deal with conclusions that the user disagrees with. 2), a system must be able to justify why it needs a particular piece of information, and 3) a system should be able to explain why it has not reached a particular conclusion, why it has not made a particular recommendation (Sell, 1985).

The above characteristics were presented to show what would be desirable in a fully functioning expert system. However, these qualities will not all be obtainable in a spreadsheet design system.

4.4 Expert System Architecture

The best reference the author found for a basic description of expert system architecture was "Expert Systems for Civil Engineers" edited by Mary Lou Maher. A condensed version of these description is presented below. "The basic architecture of an expert system exhibits a separation of domain knowledge, control knowledge, and knowledge about the specific problem currently being solved. This leads to the identification of three basic components of an expert system: the knowledge base, the context,

and the inference mechanism. Additional components needed to make the expert system more usable are a user interface and an explanation facility. (Also), to enhance extendability, a knowledge acquisition facility is desirable" (Maher, p.6, 1987).

Therefore, the architecture of an expert system is commonly described as having these six components:

(1) knowledge base

(2) context

(3) inference engine

(4) knowledge acquisition

(5) explanation system

(6) user interface

Additionally, there has to be the use of some support software (implementation language) which will provide the interface between the components, the hardware, and the user.

4.4.1 Knowledge Base

"The knowledge base is the component of an expert system that contains the inputted facts and heuristics rules associated with the domain (field) in which the expert system is applied" (Maher, p.6, 1987). An experts system's knowledge is derived from rules. The knowledge base is the form of rules and facts that are structured for the purpose of either diagnosis or consultation. The expert system cannot do the converse, that is create new rules from its knowledge. Expert systems do not currently process this ability to learn. It is important that the knowledge base be transparent (easy to read and visualize) enough so that it can be easily

modified. "Modification is especially important in most engineering domains (fields) since knowledge is continually changing and expanding" (Maher, p.7, 1987).

4.4.2 Context

"The context is the component of the expert system that contains the information about the problem currently being solved. The context initially contains the information that defines the parameters of the problem and, as the expert system reasons about the given problem, the context expands and contains the information generated by the expert system to solve it. Upon completion of the problem solving process of the expert system, the context contains all the intermediate results of the problem solving process as well as the solution" (Maher, p. 7, 1987). In other words the context is the repository for all the information concerning the current problem or current "run" of the system.

4.4.3 Inference Engine (Control Strategies)

"The inference engine is the part of the expert system that contains the control information. The inference engine uses the knowledge base to modify and expand the context" (Maher, p. 7, 1987). An inference engine controls and operates the strategies of the system. It controls the selection of rules from the knowledge base and decides when the user needs to be queried for additional information. The inference engine must be designed to use the knowledge base as structured.

4.4.4 Explanation System

The explanation component allows the user to query the system for an

explanation as to the reasoning and strategies used by the system to reach its conclusion. "The explanation can vary from only a trace of the systems execution to the ability to respond to question about the reasoning process used to develop a solution. Any well-written interactive computer program contains a trace of its execution; this is not a concept unique to expert systems. (However), an expert system can provide more than a passive trace by responding to questions about specific aspects of the problem solution" (Maher, p. 7, 1987).

It is through the explanation process that the user is presented the logic and rules of the system. This is perhaps the most important component for an engineer. It allows the engineer to access the program's logic used in arriving at the solution. The importance is obvious if the engineer disagrees with the program's solution. The explanation facility " allows the engineer to spot faulty program logic or to spot mistakes in his own logic to the problems solution. This insight into the interpretation of the results makes the expert systems program valuable to the engineering profession (Sell, 1985).

4.4.5 Knowledge Acquisition

The knowledge acquisition facility in an expert system is the component that facilitates entering knowledge into the knowledge base" (Maher, p. 8, 1987). Knowledge acquisition is that process of obtaining information and expertise from individuals familiar with the problem to be solved. "In the simplest case, this facility acts as an editor, and the knowledge is entered directly in a form acceptable by the software in which the expert system is implemented. On a more sophisticated level,

the knowledge acquisition facility understands the inference mechanism being used and can actively aid the expert in defining the knowledge base rules" (Maher, p. 8, 1987). Remember, knowledge only comes from individual experts familiar with the domain, not the system itself. Therefore, the process of reviewing and updating information is an important part of any viable system.

4.4.6 User Interface

"The expert system user interface extends the traditional capabilities of conventional user interfaces. In addition to being highly interactive, perhaps with 'HELP' facilities, an expert system's user interface needs a transparency of dialogue, whereby some form of an explanation facility indicates the inference, or reasoning, process used" (Maher, p. 8, 1987). User interface in expert systems is associated with a high degree of user friendliness and has demonstrated the value of such programming approaches for gaining user acceptance.

4.5 Problem Solving Control Strategies

The problem solving control strategies described in this section emphasizes expert systems developed in the electronic spreadsheet. However, the principles apply evenly to all expert systems.

A rule-based knowledge system consists of rule memory (knowledge base), working memory (context), and inference engine. Lotus can be used to represent ideas, rules and common sense. These rules are constructed by using the Lotus @ IF function and the Lotus's IF and LET macros clauses. An If/Then rule has the form: If (this is true), then (do this).

The IF macro rule has the form:

{if condition} {do this macro if condition is true}

{do this macro if condition is false}

A "condition" is anything that results in a true/false test (=,≠,<,≤,>,≥).

The LET macro has the form:

{let location, number or string}

The contents of the cell described by location are replaced by the number or string in the {let...} statement.

The LET clause is used as the then or the consequent of the if condition (If/Then Solutions, 1987).

4.5.1 Rules

"In a rule system, rules may appear in any order. This situation is quite different from conventional computer languages or Lotus macros where the instruction are executed sequentially (algorithm)" (If/Then Solutions, p. 11, 1987). Rules can have several clauses with several attributes and values. A rule is said to fire if its conditions are satisfied or it fails to fire if its conditions are not satisfied. One rule by itself is instructive but not very useful. To describe different aspects of a situation, many rules are needed. It is this collection of rules which forms the knowledge base in a rule based system. "Rules never communicate directly with each other. All they can do is succeed (true) or fail (false). If they fail, nothing happens; if they succeed, typically the value of one or more attributes is changed" (If/Then Solutions, p. 11, 1987), and the rule is said to have fired. Having rules and facts is not enough, a mechanism is still needed for selecting and using the rules and

facts (control strategy). This mechanism is the inference engine. The inference engine searches for rules that can fire and then allows them to preform. It does this by examining the IF part of the rule to determine if it will fire. This form of search is called "chaining". Expert systems in Lotus uses a chaining procedure for an inference engine. Rules are broken into clauses and clauses are broken into attribute pieces and value pieces. These pieces are stored in a Lotus data base to simulate the system knowledge for use in the chaining process (If/Then Solutions, 1987).

4.5.2 Rule Representation

"The behavior of rule-based systems can often be difficult to understand. One useful way to visualize their behavior is to put the rules and their linkages in a graphical form called a tree" (If/Then Solutions, p. 20, 1987). Figure 4.1 shows a single conclusion rule tree were attribute values are represented by letters.

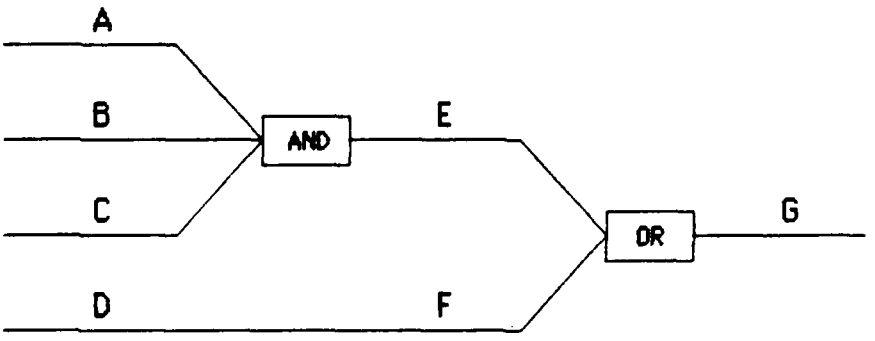


FIGURE 4.1

The rule tree in Figure 4.1 reads: If rules A, B, and C are true, then E is true; if the rule D is true, then F is true; and if either the rule E or F is true then G is true. The order in which these rules were read is an

example of forward chaining which is discussed in the following section. Note that several facts or attribute_value pairs lead into one node to make a conclusion. The node being the point where several lines (rules) in a graph come together. The rules which consist of attributes and values are represented by limbs or lines on the graph. The conclusion or THEN side of a rule (E and F) can be the input or IF side of another rule (G). Thus, facts and rules can provide linkages, creating trees that represent the rule base. The term conjunction is used to describe a rule where the clauses are joined by "and", and the term disjunction is used when the clauses are joined by or.

In some cases, instead of several IF clauses and one THEN clause, there might be one rule that reach several conclusions from the same fact or attribute_value pair. Hence, the tree has a different appearance. Figure 4.2 shows a rule tree with two conclusions.

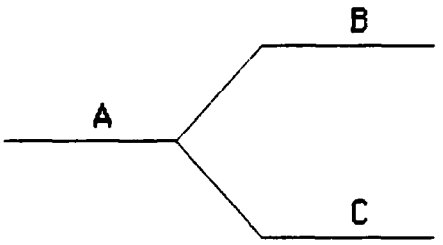


FIGURE 4.2

This tree reads: if rule A is true, then rules B and C are true. The two cases just graphed can be combined into one graph. As shown in Figure 4.3

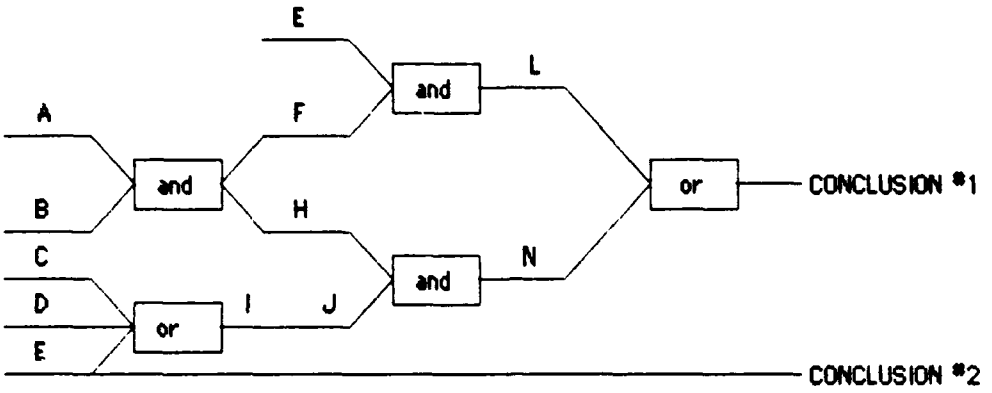


FIGURE 4.3

The attribute_value pairs at the far left end of the tree cannot be determined by the application of any rule. The value for the attribute must come from outside the rule and fact base(i.e., from user or spreadsheet results). The values for the attributes that are the THEN part of a rule can be obtained by firing a rule. Thinking about the tree representation of a rule base can give a better grasp of what happens as rule-based systems run. For example, in the forward chaining discussion in the next section, instead of attempting to visualize rules firing at random, try visualizing the propagation of rules firing from left to right
within a tree. While many strategies for solving problems exist, only the two basic strategies which are used in a spreadsheet system, are presented below (If/Then Solutions, 1987).

4.5.3 Forward Chaining

Chaining refers to the method the inference engine uses to progress through the rules of the knowledge base. A forward-chaining strategy works from the initial known facts and tries to reach a goal (conclusion or conclusions). The system is called a forward chaining system because of the way it starts at the beginning and works its way through all the

systems rules to reach the goal. The input to a forward chaining system includes the value of all the fact that the system has in its knowledge base plus the input provided by the user. "This strategy is most useful in situations where there are many hypotheses (solutions) and few input data. Its main drawback is that it may be wasteful to require as input all the possible facts for all conditions; in many circumstances all possible facts are not known or relevant" (Maher, p.11, 1987). Forward chaining systems have been most appropriate for tasks in the general category of planning (If/Then Solutions, 1987).

The simplest way to visualize forward chaining is from a rule tree. Visualize the inference engine searching from rule to rule through the tree. When a rule fires, the values of the atributes are changed. As a result, any rules that previously were unable to fire now may be able to. The searching continues until all rules that can fired have been fired. Hopefully, then the rules will be able to provide a conclusion about the problem's input. A forward chaining logical rule tree is shown in figure 4.4.

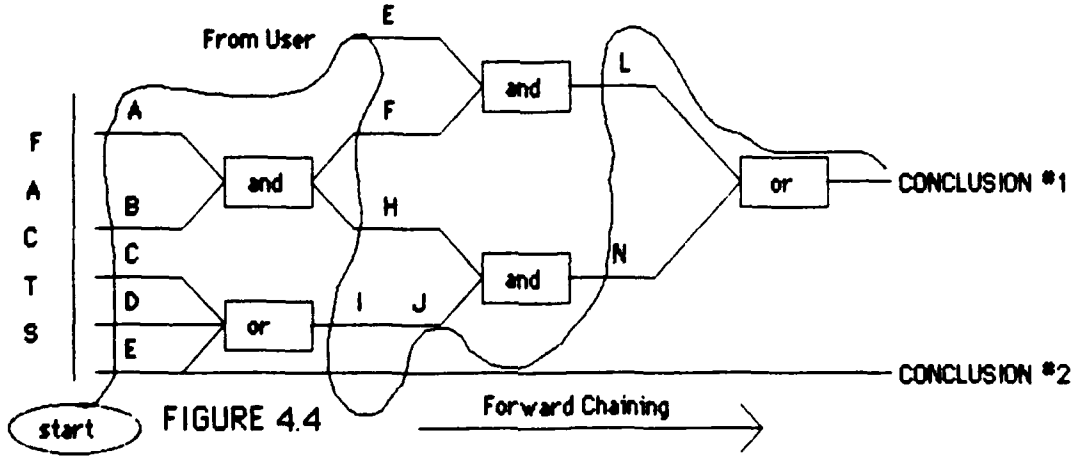


FIGURE 4.4

In the forward chaining example in Figure 4.4, the inference engine starts

with the initial facts, i.e., user inputs or spreadsheet results, and progressively draws conclusions from the system's rules. To perform the forward chaining operation the system evaluates all the rules who can fire. The advantages are that the logic is simple to follow, and if the user is familiar with Lotus macros, the system is easy to construct (If/Then Solution, 1987).

4.5.4 Backward Chaining

A backward-chaining strategy assumes a goal, hypothesis, or conclusion and works backward to known data or facts to support or discount the assumed conclusion, goal, or hypothesis. The process continues until a conclusion is reached. Sometimes, no conclusion is reachable from the given facts and rules. In backward chaining, the input information sorted by the inference engine is limited to the minimum amount needed to solve the problem. "If the known facts do not support the hypothesis, then the preconditions that are needed for the hypothesis are set up as sub goals. This process continues until the original hypothesis is either supported or not supported by known facts" (Maher, p. 12, 1987). The order in which the hypotheses are tested is predefined in the knowledge base and inference engines. A backward chaining logical rule tree is shown in Figure 4.5. The example of the rule tree in Figure 4.5 for the backward chaining looks the same as for forward chaining tree, except for the path that is followed to prove or disprove the conclusion. The chaining begins with a conclusion and trys to prove it true by working backwards through the tree. If the backward chaining can not prove the conclusion true from the given rules and facts, it can ask the user for

additional information needed to reach a conclusion.

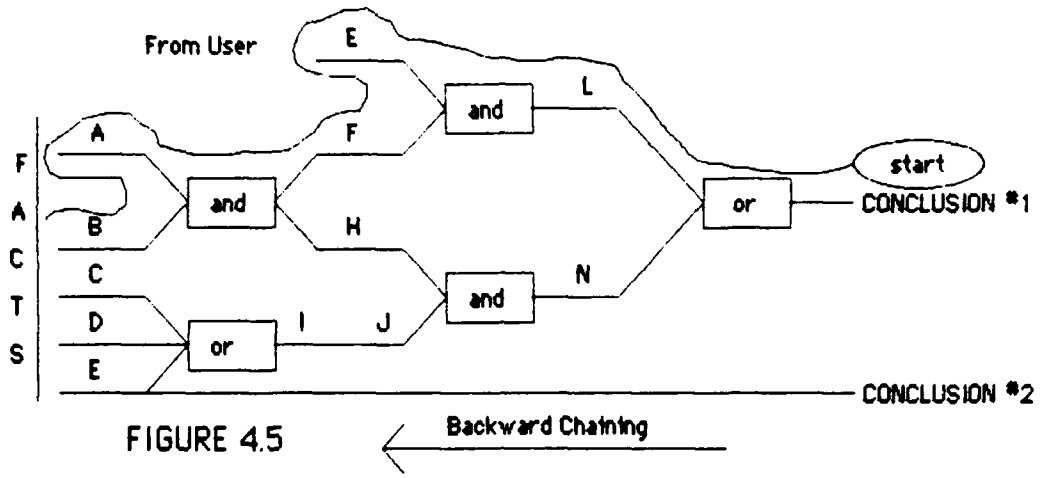


FIGURE 4.5

The advantage of this type of control strategy is that only pertinent information to the problem is required for a solution. This can save time and makes things easier on an end user who many have a difficult time finding information about irrelevant parameters (If/Then Solutions, 1987).

4.5.5 Mixed Chaining

Often a system can use a mixed strategy by combining the forward-chaining and backward-chaining strategies. In the mixed initiative strategy, the order in which the rules are checked depends on the problem under consideration and the program's design. "The advantage of this strategy is that the user supplies only the data relevant to the problem at hand and, if an initial hypothesis is disproved, the next assumption is made according to current information" (Maher, p. 12, 1987). This type of chaining incorporates the positive features of ease of understanding from the forward approach and efficiencies from the backward chaining approach.

4.6 Expert System Shells

Shells are generically formatted expert systems which do not contain any specific knowledge base rules.

IF/THEN provides some simple "expert system shells" in the form of Lotus templates. These templates contain the inference engine and the working memory. The user only provides the rules (knowledge base). The use of these shell templates is a practical way to start the development of a rule base expert system.

Shells have several notable advantages. Many shells are inexpensive and can be run on widely available microcomputers. Often, only a moderate time investment is needed to learn how to use a shell productively, even for persons having limited experience with computers. Because shells make it possible to build experimental prototypes quickly and without extensive programming, they are convenient for "rapid prototyping".

Restrictions of shells are (Ortolano, Perman, 1987):

1. Rigidities in the inference engines. Control is often restricted to either forward or backward chaining methods. The procedures for deciding which rules to fire or when to halt the search process is often not controllable by the system developer.
2. Restraints in knowledge representation. Shells typically restrict the way rules can be represented. This can lead to awkward or excessively complex rule statements. Sometimes a large number of rules must be used to represent a small set of ideas.
3. Limitations of explanation facilities. Explanation facilities are

often inflexible and sometimes yield output explanations that are difficult to understand or not provided at all.

4.7 Summary

Human experts solve problems in many ways. They can use an algorithmic approach composed of a sequence of action, a trial and error method, or knowledge about the situation to form constraints that can lead to the problems solution. An expert system tries to duplicate the method of problem solving which uses specific knowledge about the situation to form constrains which will lead to some conclusion. An expert system is defined as a knowledge-based system that emulates expert thought to solve significant problems in a particular domain of expertise. It is important to note that the manipulation of the expert system's knowledge-base is usually carried out by a computer, but there is no requirement that an expert system has to be a computer program; it is just faster, more accurate, and more convenient to implement it as a computer program (Sell, 1985).

The aim of this chapter was to examine how expert systems operate and how their techniques can be implemented within a Lotus spreadsheet program. An account of expert system fundamentals was presented. The term "expert system" was defined; knowledge representation criteria and the rule based system were discussed; the most desirable characteristics and the basic architecture of expert system were defined; and the most popular form of control strategies were previewed.

Some of the mystery of expert systems has hopefully been removed. Expert systems are neither unusual nor inherently difficult to understand.

In fact, there is nothing in expert system development in which someone with a reasonable understanding of the expert system components and a firm grasp of the problems solution method could not reproduce as a small scale expert system once shown how to start.

CHAPTER FIVE
CONCLUSION

This report has reviewed the engineering application of the computer spreadsheet and has looked at the basic ideas and concepts of expert system technology application.

Hopefully, it has led to the conclusion that a vast majority of engineering work can be accomplished on the computer spreadsheet and that an expert systems is just an advancement of conventional computer programming techniques.

The computer spreadsheet helps reduce the "black box" effect for technical calculations. It provides on screen documentation of the problem's solution and the solution method. With the spreadsheet's good response time, sensitivity analysis is easy. The time required to learn and operate a computer spreadsheet is shorter than learning and operating traditional programming with its array of input and output statements. The presentation of results using the spreadsheet is amazingly simple and readily presentable. Adding to the above the advance programming and data base function, a powerful engineering tool is available for the micro computer.

Small scale development of expert systems in computer spreadsheets will not require a"professional" programmer to design them. With the basic ideas and concepts of expert systems, the spreadsheet user can incorporate these new techniques.

A spreadsheet based expert system will never be complete enough to sacrifice the engineer's common sense or engineering judgement.

However, as computer programs and usage become more automated the chance of data to be transmitted or translated erroneously is increased. Solution of this problem will provide a good use for spreadsheet based expert system techniques which can check the data entry and data base records for errors and call the results to the attention of the engineer. The technique is capable of assisting the engineering.

Before development of an expert system is undertaken, it requires complete domain knowledge plus the programming knowledge. Then starting with a simple situation, define the attributes and values, and slowly build and experiment as you gradually expand. Expert systems may be costly to develop and limited in their ability to solve problems. They become quite complicated and hard to maintain. While the techniques are useful, it is not the final solution.

Finally, the micro computer has greatly expanded the engineer's effectiveness; however, the engineer should only use the micro computer as a powerful tool and not allow his judgement to be sacrificed. The "black box" syndrome, infinite accuracy, data input integrity, instant expertise, and lack of judgement are all potential misuses of the micro computer.

APPENDIX I

Appendix I is a sample electronic spreadsheet program which calculates the optimum time to replace equipment. The original version of the program comes from the second edition, Engineering Economic and Cost Analysis by Courtland A. Collier and William B. Ledbetter.

	A		B	C	D	E	F	G	H	I
1	**Defender/Challenger, w/ taxes, income & inflation**									
2	This template finds the optimun retirement time for the Defender, and the optimum life cycle time for the replacement Challenger									
3	Note: All Defender payments are postscripted with a "D", while all Challenger payments are postscripted with a "C".									
4	**Input Data, General**									
5	Owner's After Tax MARR,	i =	15.00%	per year (program assumes r8 < i & r10 < i)						
6	Inflation Rate,	r1 =	5.00%	per year						
7	Owner's Income Tax Rate,	t =	28.00%	per year						
8			**Defender**		**Challenger**	units				
9	Cost New Listed Price,	P1D=	14,135	P1C=	14,700	$				
10	Fleet Discount Rate	r2D =	12.00%	r2C =	12.00%					
11	Defender's Loan Balance		8,353			$				
12	Challanger's Loan /Value ratio			r3C =	80.00%					
13	Loan Rate	r4D =	10.00%	r4C =	10.90%	per year				
14	Loan Term (remaining)	LnD =	3	LnC =	4	years				
15	Purchase Trans. Cost rate	r5D =	7.00%	r5C =	7.00%					
16	Resale Value @ PTZ,	P2D=	11,300			$				
17	Decline in Resale Value,	r6D=	-20.00%	r6C =	-20.00%	yr/yr				
18	Resale Transaction Cost	r7D =	5.00%	r7C =	5.00%					
19	Repair Cost @ EOY 1,	A1D=	150	A1C =	100	$/yr				
20	Arith Grad Repair Cost,	G1D=	50	G1C =	50	$/yr/yr				
21	Periodic Repr Cost @EOY5,	F1D =	1,200	F1C =	1,260	$				
22	O&M Cost @ EOY 1,	C1D=	4,000	C1C=	4,000	$/yr	(Geometric Gradient)			
23	O&M Cost Proj. Increase,	r8D=	5.00%	r8C =	5.00%	yr/yr	(assumes r8 < i)			
24	Depreciation Life,	DLD=	3	DLC =	3	years				
25	Defender's Present Age,	ageD=	1			years				
26	Daily Rental Income	DRID =	34.00	DRIC =	36.00	$/day				
27	Utilization rate,	r9D =	260	r9C =	274	day/yr				
28	Decline in Utilization rate	r10D	-5.00%	r10C =	-5.00%	/yr/yr	(assumes r10 < i)			
29	Income Penalty for age,	IPD =	-2.00	IPC =	-2.00	$/dy/yr-age				
30	Arith. Grad. Income,	G2D=	-520	G2C=	-548	$/yr/yr	= IP * r8			
31	Annual Income @EOY,	C2D=	8,840	C2C =	9,864	$/yr	= DRI * r8 (Geometric Gradient)			

	A	B	C	D	E	F	G	H	I
32	In the matrix below, the PNW calucated for the Challenger at PTZ (N = 0) is escalated from the value at the PTZ to the higher								
33	replacement value at EOY N by multiplying by (F/P, r1, N), where r 1 is the price escalation rate. Then the PW of this escalated								
34	value is found at PTZ by multiplying by (P/F, i, N), where i is the investment rate. Then the Defender's NPW is added to the								
35	Challenger's NPW for a total NPW for the remaining Defender's life plus the infinite series of Challenger replacement cycles								
36	**Matrix,ChallengerLife, L-->**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
37	**Defender Life, N/** **0**	1,085	-506	-2,084	-3,782	-2,298	-2,234	-1,031	274
38	**1**	1,090	-362	-1,802	-3,353	-1,998	-1,940	-841	350
39	**2**	801	-525	-1,840	-3,256	-2,019	-1,966	-962	125
40	**3**	214	-997	-2,198	-3,491	-2,361	-2,312	-1,396	-403
41	**4**	-755	-1,860	-2,956	-4,137	-3,106	-3,061	-2,225	-1,318
42	**5**	-1,909	-2,918	-3,919	-4,997	-4,055	-4,015	-3,251	-2,423
43	**6**	-3,636	-4,558	-5,472	-6,456	-5,596	-5,559	-4,862	-4,106
44	**7**	-5,037	-5,878	-6,712	-7,611	-6,826	-6,792	-6,155	-5,465
45	The optimum lives are found by locating the highest NPW in the matrix. In this example the highest NPW is +$1,084 found when N = 1,								
46	and L = 1. This indicates that the Defender should be retained in service for 1 more years, and then replaced with the Challenger								
47	which will have a 1 year life cycle. The cost penalties for early or late replacement for the Defender are found by comparing the								
48	figures that are adjacent in the matrix. For instance replacement one year early would cost $6 in terms of NPW (subtract the								
49	$1,084 value at N = 0, L= 1 from the $1,090 optimum figure). In similar manner the penalty for replacement one year late is								
50	found as $289 (in terms of NPW).								
51									
52	**Defender's Equations**								
53		AMORTIZATION SCHEDULE					PRIN+INT(1-t)		
54	Defefender Life @ PTZ, N =	PMT	INT	PRIN.	BAL	INT(1-t)	(P/F,i,N)	Σ	Bal(P/F,i,N)
55	0				8,353		0	0	-8,353
56	1	3,359	835	2,524	5,829	601	-2,717	-2,717	-5,069
57	2	3,359	583	2,776	3,054	420	-2,416	-5,134	-2,309
58	3	3,359	305	3,054	0	220	-2,152	-7,286	0
59	4	0	0	0	0	0	0	-7,286	0
60	5	0	0	0	0	0	0	-7,286	0
61	6	0	0	0	0	0	0	-7,286	0
62	7	0	0	0	0	0	0	-7,286	0
63									
64	Defender's Trial Life, years, N=	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
65	Depreciation:								
66	Depreciationed Book Value,$, SLBVD=If(ageD+N<DLD,then SLBVD=P1D(1-r2D)(1+r5D)[DLD - (ageD + N)]/DLD)orIf(ageD+N>=DLD,then SLBVD = 0)								
67	SLBVD =	8,873	4,437	0	0	0	0	0	0
68	Remaining Years of Depreciation, N2 = IF(ageD + N) ≤ DLD, then N2 = N. Or if not , then N2 = DLD - ageD =								

	A	B	C	D	E	F	G	H	I
69	N2 =	0	1	2	2	2	2	2	2
70	PW of the Aftertax Depreciation Credit @ PTZ, $, P3D = [(P1D/DLD)*t]*(P/A,i,N2)								
71	**P3D =**	0	1,080	2,019	2,019	2,019	2,019	2,019	2,019
72	Equivalent Cash Invested in Defender:								
73	Cash Investment in Defender @ PTZ,after taxes, $, P4D = P2D*(1-r4D) - (P2D*(1-r4D)-SLBV)*t								
74	**P4D =**	-10,214	-10,214	-10,214	-10,214	-10,214	-10,214	-10,214	-10,214
75	PW of Loan Balance @PTZ, after taxes, $, P5D = (loan balance @ EOY N)*(P/F,i,N)								
76	**P5D =**	-8,353	-5,069	-2,309	0	0	0	0	0
77	PW of Loan PMTs (AT.INT + PRIN.) @PTZ, after taxes, $, P6D = Σ[(PRIN. + INT*(1-t)(P/F,i,1)+ . . . +(PRIN. + INT*(1-t)(P/F,i,L)]								
78	**P6D =**	0	-2,717	-5,134	-7,286	-7,286	-7,286	-7,286	-7,286
79	Future Resale Value:								
80	@ EOY N, after taxes, $, F2D = P2D*(1+r5D)^N*(1-r6D)*(1-t)+SLBVD*t								
81	F2D =	10,214	7,426	4,947	3,957	3,166	2,533	2,026	1,621
82	Present Worth @ PTZ, after taxes, $, P7D = F2D*(P/F,i,N)								
83	**P7D =**	10,214	6,457	3,740	2,602	1,810	1,259	876	609
84	Repair Cost with Arithmetic Gradient:								
85	PW of Repair Costs @ PTZ, after taxes, $, P6D = A1D*(P/A, i, N)*(1-t)								
86	**P8D =**	0	-94	-176	-247	-308	-362	-409	-449
87	PW of Repair Gradient @ PTZ, after taxes, $, P7D = G1D*(P/G,i,N)*(1-t)								
88	**P9D =**	0	0	-27	-75	-136	-208	-286	-367
89	Periodic Over Haul Repair Performed @ EOY 5:								
90	PW of Periodic Repair Cost @ PTZ, after taxes, $, P10D = IF(N=<5 then P10D = 0) or IF (N > 5 then P10D= F1D*(P/F,i,5)*(1-t))								
91	**P10D =**	0	0	0	0	0	0	-430	-430
92	O&M Costs with Geometric Gradient:								
93	w =[(1+i))/(1+r8D)]-1 =	0.0952381	(assumes r8D < i)						
94	PW of O&M Geometric Gradient Cost @PTZ, after taxes, $, P8D = C1D*(P/A,w,N)/(1+r8D)*(1-t)								
95	**P11D =**	0	-2,504	-4,791	-6,879	-8,785	-10,525	-12,114	-13,565
96	Income with Arithmetic And Geometric Gradient:								
97	w=[(1+i)/(1+r10d)]-1 =	0.21052632	(assumes r10D < i)						
98	PW of Income @PTZ, after taxes, $, P9D = C2D*(P/A,w,N)/(1+r10D)*(1-t)								
99	**P12D =**	0	5,535	10,107	13,884	17,004	19,581	21,710	23,469
100	PW of Income Gradient @ PTZ, after taxes, $, P10D = G2D*(P/G,i,N)*(1-t)								
101	**P13D =**	0	0	-283	-775	-1,418	-2,162	-2,972	-3,816
102	Total-Net Present Worth								
103	**NPW of Defender, after taxes, $, NPWD = P3D+P4D+P5D+P6D+P7D+P8D+P9D+P10D+P11D+P12D+P13D**								
104	**NPWD =**	-8,353	-7,527	-7,067	-6,970	-7,314	-7,897	-9,104	-10,029

	A	B	C	D	E	F	6	H	I
105	**Challenger's Equations**								
106	In this section, all Challenger equations assume that the remaining Defender life, N, is zero (Challenger replaces Defender right now).								
107	Cost new of the replacement Challenger, plus a infinite series of subsequent Chanllenger replacements.								
108	To find PW, the cost is first transferred to the end of the Challenger's life (@ EOY L) for convenience in using								
109	the infinite geometric P/C series equation to find the PW.								
110	**AMORTIZATION SCHEDULE: Loan Balance = P1C*(1-r2C)*(1+r5C)*(r3C)**						PRIN+INT(1-t)		
111	Challenger's Life L=	PMT	INT	PRIN.	BAL	INT(1-t)	(P/F,i,N)	Σ	Bal(P/F,i,N)
112	0				11,073	0	0	0	-11,073
113	1	3,562	1,207	2,355	8,719	869	-2,803	-2,803	-7,581
114	2	3,562	950	2,611	6,107	684	-2,492	-5,295	-4,618
115	3	3,562	666	2,896	3,212	479	-2,219	-7,514	-2,112
116	4	3,562	350	3,212	0	252	-1,980	-9,495	0
117	5	0	0	0	0	0	0	-9,495	0
118	6	0	0	0	0	0	0	-9,495	0
119	7	0	0	0	0	0	0	-9,495	0
120	8	0	0	0	0	0	0	-9,495	0
121									
122	Callenger's trial Life Span, L=	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
123	Eff. Int. Rate, ie = (1+i)^L-1 =	0.15	0.3225	0.520875	0.74900625	1.01135719	1.31306077	1.66001988	2.05902286
124	Eff. Esc. Rate, re =(1+r1)^L-1 =	0.05	0.1025	0.157625	0.21550625	0.27628156	0.34009564	0.40710042	0.47745544
125	Geo. Grad. we=(1+ie)/(1+re)-1=	0.0952381	0.19954649	0.31378901	0.43891177	0.57595099	0.72604156	0.89042647	1.07046708
126	(P/C, we, ∞)=1/[(1+re)*we] =	10	4.54545455	2.75292498	1.87441425	1.36040424	1.02778607	0.7981359	0.63228414
127	(F/P, i, L) =	1.15	1.3225	1.520875	1.74900625	2.01135719	2.31306077	2.66001988	3.05902286
128	(F/A,i, L) =	1	2.15	3.4725	4.993375	6.74238125	8.75373844	11.0667992	13.7268191
129	Cash Invested in Challanger:								

	A	B	C	D	E	F	G	H	I
130	PW of Purchase Price +∞ series of replacements, @PTZ, $, PC2 = -[P1C*(1-r2C)(1+r5C)]*(F/P,i,L)*(P/C,we,∞)								
131	**P2C =**	-159,177	-83,206	-57,952	-45,378	-37,874	-32,906	-29,386	-26,772
132	PW of Loan, after taxes, @PTZ, $, P3C = [[Amt. Borrowed]-[Ln.bal@eoy L](P/F,i,L)-Σ[(PRIN+INT(1-t))(P/F,i,L)]]*(F/P,i,L)*(P/C,we,∞)								
133	**P3C =**	7,920	6,974	6,060	5,175	4,320	3,753	3,352	3,053
134	Depreciation:								
135	The depreciation credit is cut off at the end of the Challenger's depreciation life (DLC). If the trial L value <= DLC,								
136	then F2C = (F/A,i,L). If the challanger's trail life (L) value is> DLC, then F2C = (F/A,i,DLC)*(F/P,i,L-DLC).								
137	F2C =	1	2.15	3.4725	3.993375	6.60373844	10.2543191	15.3103432	22.2592861
138	Annual Depreciation credit, after tax, $/yr, A1C = ([P1C*(1-r2C)(1+r5C)]/DLC)*t =					1,292			
139	PW of the Aftertax Depreciation Credit @ PTZ, after taxes, $, P4C = A2C*F2C*(P/C,we,∞)								
140	**P4C =**	12,919	12,625	12,350	9,670	11,606	13,615	15,786	18,182
141	Depreciationed Book Value, $, SLBVC = If(L<DLD then SLBVC=P1C*(1-r2C)*(1+r5C)*[DLD - L]/DLD ,or, (If L>=DLC, then SLBVC = 0)								
142	SLBVC =	9,228	4,614	0	0	0	0	0	0
143	Future Resale Value:								
144	@ EOY L, after taxes, $, F3C = P1C*(1-r7C)*(1+r6C)^L*(1-t)+SLBVC*t =								
145	F3C =	10,628	7,727	5,148	4,118	3,295	2,636	2,109	1,687
146	Present Worth @ PTZ, after taxes, $, P5C = F2C*(P/C,we,∞)								
147	**P5C =**	106,276	35,122	14,172	7,720	4,482	2,709	1,683	1,067
148	Repair Cost with Arithmetic Gradient:								
149	PW of Repair Costs @ PTZ, after taxes, $, P6C = A1C*(F/A, i, L)*(P/C,we,∞)*(1-t)								
150	**P6C =**	-720	-704	-688	-674	-660	-648	-636	-625
151	PW of Repair Gradient @ PTZ, after taxes, $, P7C = G1C*(F/G,i,L)*(P/C,we,∞)*(1-t)								
152	**P7C =**	0	-164	-312	-447	-569	-679	-779	-869
153	Periodic Over Haul Repair Performed @ EOY 5:								
154	PW of Periodic Repr Cost @PTZ, after taxes, $, P10C= IF(L=<5 then P10C=0)orIF(L>5 then P10C= F1C*(P/F,i,5)(F/P,i,L)(P/C,we,∞)(1-t))								
155	**P8C =**	0	0	0	0	0	-1,072	-958	-872
156	O&M Costs with Geometric Gradient:								
157	w =[(1+i)/(1+r8C)]-1 =	0.0952381	(assumes r8C < i)						
158	PW of O&M Geometric Gradient Cost @PTZ, after taxes, $, P9C = C1C*(P/C,w,L)*(F/P,i,L)*(P/C,we,∞)*(1-t)								
159	**P9C =**	-28,800	-28,800	-28,800	-28,800	-28,800	-28,800	-28,800	-28,800
160	Income with Arithmetic and Geometric Gradients:								
161	w =[(1+i))/(1+r10C)]-1	0.21052632	(assumes r10D < i)						
162	PW of Income @ PTZ, after taxes, $, P10C = (C2C*(F/A,i,L))*(P/C,we,∞)*(1-t)								
163	**P10C =**	71,021	67,793	64,862	62,201	59,785	57,591	55,598	53,788
164	PW of Income Gradient @ PTZ, after taxes, $, P11C = G2C*(F/G,i,L)*(P/C,we,∞)*(1-t)								
165	**P11C =**	0	-1,793	-3,422	-4,898	-6,235	-7,445	-8,538	-9,525
166	Total-Net Present Worth:								
167	**NPW of Challenger, after taxes, $, NPWC = P2C+P3C+P4C+P5C+P6C+P7C+P8C+P9C+P10C+P11C =**								
168	**NPWC =**	9,438	7,847	6,269	4,571	6,055	6,119	7,322	8,627

APPENDIX II

Appendix II is a summary of the South Florida's Surface Water Management - prototype decision support system for reviewing surface water permit application. The prototype system is being developed under contract by Dr. James Heaney, Professor, Environmental Engineering Department of the University of Florida.

The purpose of this appendix is to provide further information on the prototype spreadsheet based system discussed in Chapter Three. As stated, this prototype system represents state of the art use of electronic spreadsheets for the solution of engineering problem. The information in this appendix is a condensed summary of the prototype system and its progress report (Heaney, et al., 1988). It is not the author's original work nor is it an any way presented to be so. The author's involvement with this project's development was only as an observer.

With the basic concepts of the electronic spreadsheet and expert systems described herein this report, the following information is presented to help the reader fully understand and appreciate the powerful consequences of the current micro computer spreadsheet technology and to view the practical application of these concepts.

Surface Water Application Permit Review - Decision Support System

This project is the current state of the art in the engineering use of computer spreadsheets. The prototype system is part of a project funded by the South Florida Water Management District. The project is being developed under Dr. James Heaney, Professor, Environmental Engineering Department of the University of Florida.

Since this is an on going research and a prototype development project, this appendix summary is limited to the development of the project through the summer of 1988 (phase 1). The information presented in this chapter is taken from a draft progress report of the project prepared in August 1988 (Heaney, et al., 1988) and from hands-on experience gained in operating the prototype system which consists of the readily available commercial software applications of Lotus's spreadsheet, @Base's data files, and WordPerfect's wordprocessing files.

Surface Water Management Project

The primary emphasis of the development project has been the creation of a data based spreadsheet system oriented to meet the needs of a review engineer in the surface water management review process. The integration of commercially available software such as spreadsheets,

wordprocessors, and data base management has allowed for flexibility in the system's prototyping.

The goals of the project were to develop, provide, and test a spreadsheet system that would meet the surface water management's reviewing engineer needs. These needs were viewed to be a user-friendly stand alone PC-based system which would assist in the evaluation of permit applications. The elements were specified in the original request for proposal:

a) The District's Technical guidelines for conducting surface water studies (SFWMD, 1986). The guidelines should be built into spreadsheet calculation tables.

b) A knowledge base that includes permit information from the District's various data sources. This will provide the surface water permit reviewer with information regarding the site's physical characteristics (soils, ground water,elevation, assumed drainage patterns, etc.), adjacent site information (wetland areas, wellfields, etc.), local government restrictions (zoning, land use and flood plain restrictions, etc.), etc.

c) An interface with a word processor to provide the permit reviewer with the standard staff reports for the reviewed project.

d) The spreadsheet model should be able to be used as a pre- and post-processor to larger surface water models available on the CDC Cyber computer.

e) An expert system for evaluating surface water permits. This expert system would integrate the spreadsheet model and

knowledge base.

The proposal elements a through c reflect the current development of the project (phase 1). Elements d and e are to be completed later (phase 2).

The next two sections describe the surface water management review process and the developed system.

Overview of Surface Water Permit Review Process

To understand, appreciate, and comprehend the scope of the projects, a quick overview of the South Florida, Surface Water Permit process is required. This section is intended only to give the reader a grasp of the review process that the prototype system is designed to handle.

The South Florida Surface Water Management District's engineers are responsible for many activities (permit application) and decisions that affect the outcome of a multitude of South Florida developments. The first requirement in understanding the review process is to identify the participants and a definition of the relationships among them. In the case of the surface water permit review process, the primary participants are the applicants (or their engineers) and the Surface Water Management Division (SWM) staff. The primary relationship of the applicants to the review process can be seen in Figure 2-1. This relationship can be characterized as a flow of information between the reviewers and the applicant.

Review engineers conduct pre-application meetings, determine the completeness of applications, send the appropriate letters/requests to the applicants, initiate and receive telephone requests for information, assess

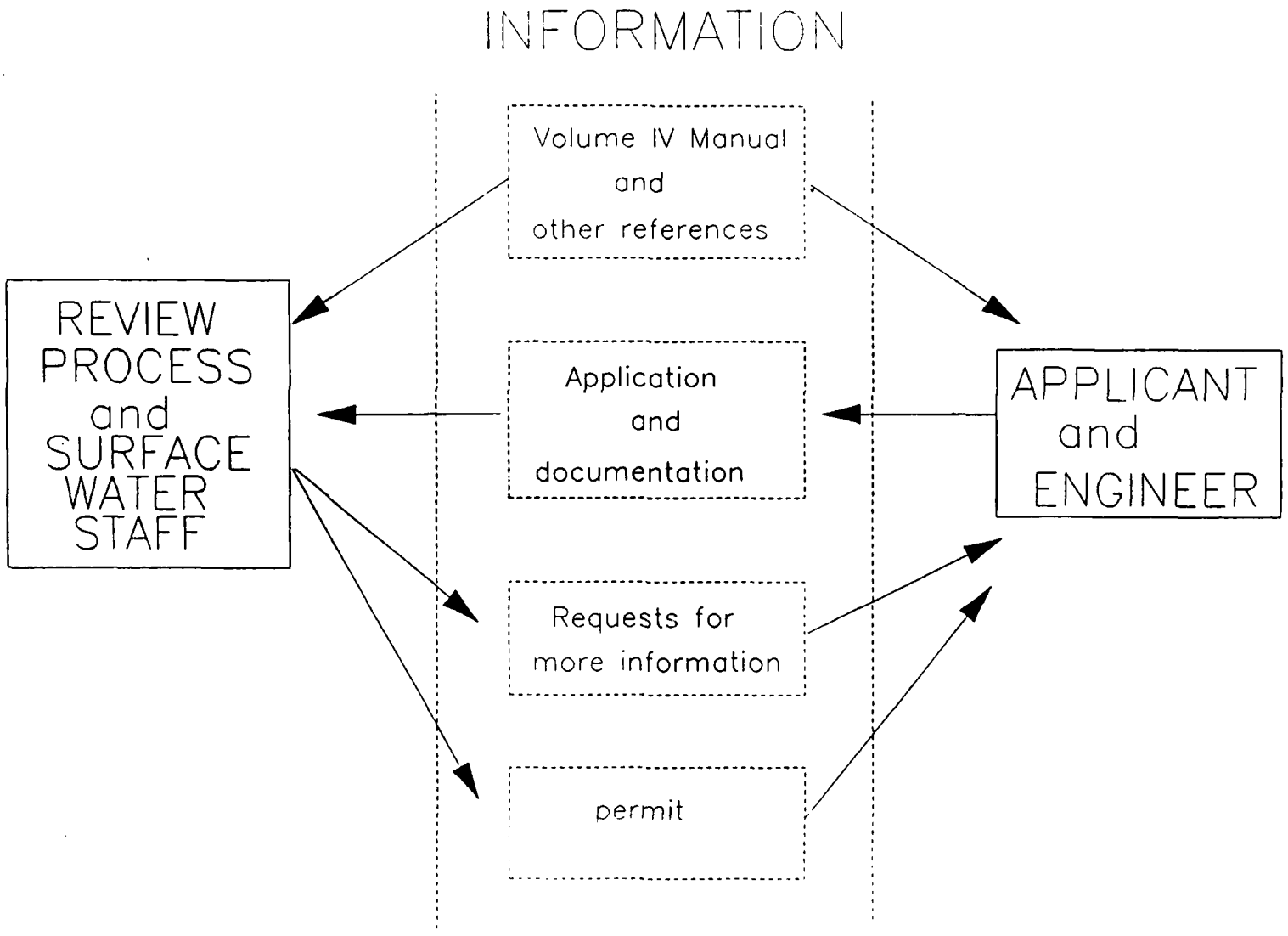


Figure 2-1: Model of Information Flows Between Surface Water Permit Applicant and Review Process.

technical design information and procedures, and compose the final staff report. Under normal circumstances, a review engineer is responsible for 20 to 30 active applications. Because of strict legal time requirements for processing applications and the number of applications under review at any one time, a clear picture of the status of individual applications must be maintained if a non "crisis mode" prioritization of work is to be maintained.

The District by law has 30 days after receiving an application to perform a sufficiency review and notify the applicant of the results. Failure of the District to respond within the 30 days requires the application to be accepted as completed.

In the sufficiency review, the SWM engineer checks the application for the required information and usually does a preliminary technical review to identify any major problems or insufficiencies in the technical data. Upon completion of the sufficiency review by the SWM reviewer, either a completeness or a 30 day (incompleteness) letter is sent to the applicant. This response and evaluation cycle continues until the application is determined to be complete. When the response is found to be sufficient, the application is viewed as being complete on the date the response was received and the "clock" governing the technical review phase is considered to have started.

The District, again by law, then has 60 or 90 days, for general and individual permits respectively, to conduct a full technical review and take final action. If no action is taken within the required time frame, the application is considered approved.

Based on the different application/permit types and the review

process, a number of possible paths/branches of the review phases were identified. The number of the different classes of application that a reviewer has to face in shown in Table 2-4. The number of classes is the product of possible application/permit types and the branches of the review process. The total number is computed to be 7,680 classes.

The primary source of data in the review process is the permit application. The permit application forms (RC-1 and RP-63) and accompanying documentation provide the data which must be evaluated. The application is reviewed from rules taken from Volume IV of the Permit Application Manual (SFSWMD 1986) and its references. Data from past permit applications of similar types or locations may also be used to appraise the suitability of the data used in the application being reviewed.

A summary of data sources that may be used in the course of a review includes the following sources:

1) Permit application and documentation
2) Permit Information Manual IV and its references
3) Parameters and data used in past applications
4) SFWMD technical reports and basin studies
5) SCS reports, DOT drainage manuals
6) Maps-- including general USGS quads, flood insurance elevation maps, land use and zoning maps
7) Textbooks and the technical literature

This completes the overview of the surface water permit review process. As can be seen, with the stringent time deadlines, the number of application which are handled, the fact that each application process is different, and the large amounts of available data and rules placed upon a

Table 2-4: Estimate of Number of Alternative Classifications of Applications in SurfaceWater Management Review Process.

	Branches
Permit Form (new, modified)	2
(indiv: concept/const/oper) general: const/oper)	5
Environmental/Water Quality Division Priority (emergency ?)	2
Basin Location (Western C-9, C-51, Kissimmee or other)	4
Project Type (resid.comm,indst,agri)	4
Discharge governed by (Pre/Post,Formula, Other)	3
Application Sufficient? (y or n)	2
Technical Complexity (single basin, multi basin)	2
Final action (denied or approved)	2
product =	7680 Classes

single review engineer, the review process needs the management and technical capabilities of the micro computer. The next section describes how the management and technical review is addressed by the prototype system.

Conceptual Overview of a Prototype Permit Review System

In this section a conceptual look at the prototype permit review system is presented. The intent of the section is to show "state of the art" spreadsheet programming in the engineering field and how the prototype system addresses the problems outlined in the previous two sections.

Within the framework of the system for reviewing surface water management permit applications the prototype system can store and retrieve application data, perform all necessary design calculations, and generate the paperwork required in the review process. The system streamlines the review process by eliminating duplication of effort on the part of a reviewer during the evaluation of an application.

As explained previously the permitting process requires a reviewer to determine an application's sufficiency, to do technical analyses, and to generate documentation of the review results. The project uses a combination of commercially available software packages versus an expert system shell software (discussed in Chapter Four) because of the mixture of checking, analysis, and documentation capabilities required in the course of a review. Also, these packages were chosen on the basis of perceived user friendliness, probability of acceptance, and ease of prototyping the system components into separate modules which all

offered flexibility in the development.

Lotus acts as the primary interface with the reviewer and provides technical analysis capabilities (design calculations) as well as the reporting and checking functions. Additionally, the Lotus add-in @Base is used for the database management of the permit review system as it is anticipate to grow quite large.

The relationship between the applicant and the review process was modeled in terms of information flows (see Figure 2-1). Expanding this model to include the required functions of the review engineer and introducing a database structure to store the information flow from the applicant and other information sources, resulted in the creation of a conceptual model for the review system. This model, illustrated in Figure 3-1, divides the reviewer's functions into four components: application tracking, checking, technical analysis, and reporting.

Each conceptual component of the review system has been assigned general responsibilities which may require the development of one or more spreadsheet templates or use of one or more of the software packages. The remainder of this section will describe the desired general structure and responsibilities of each of these components.

The database component stores all of the information required to analyze and document a permit application. The database component includes database files and the spreadsheet templates used for data entry and editing of these files. User friendliness and flexibility as well as data integrity assurance are functional requirements of the templates. Also, the database component includes any files in which application data is stored or through which application data is transferred between the other

Table 2-3: Basis for and Actions Resulting from Major Decisions in Review Process.

DECISION	BASIS	RESULTING ACTION
1) Assignment of Application to Reviewer	Project Location Reviewer Workload Experience Familiarity w/ Engineer	Reviewer(s) Assignment by Supervisor
2) Criteria Requirements of Natural Resource Management, and Water Use Divisions	Division Standards	Completed Interdepartmental Forms and Comments
3) Is the Information Provided with the Application Sufficient for a Complete Review?	Checklists Basis of Review Basin Studies Laws & SFWMD Rules Rules of Thumb	30 Day Letter, Additional Information Letter, Completeness Letter
4) Does the Application Meet the Technical Requirements of the District?	Basis of Review Technical Refs. Basin Studies Laws & SFWMD Rules Rules of Thumb	Staff Report
5) Should a Permit be Issued?	Staff Report Board Rulings	Permit or Denial

components of the system.

The tracking component searches the database for information on application assignments and review deadlines. Rules pertaining to the operation of the review "clock" are embodied in the tracking component. The tracking component updates review deadlines contained in the database.

The checking component aids the reviewer in the determination of the acceptability of the sufficiency and technical review. Design capabilities of this component include the ability to evaluate the application in the following areas:

1) application type qualifications
2) checklist guidance and summary of deficiencies
3) comparison of applicant and SWM staff technical analysis results

The formal and informal rules governing the above areas are the basis for the checking component capabilities. The checking component accesses the review databases for application specific data and refers to these rules and data internal to the component when in operation.

The technical analysis component performs design calculations required in the technical review of an application. Design calculation templates that draw their input data from and summarize their output to the database component are the basic sub units of the component. The major requirement of the technical analysis component is the ability to interface these design models with the database. The technical analysis component design will ultimately include the ability to interface the review databases in order to format input data, and receive output from

models which cannot be written in the spreadsheet environment.

The reporting component creates review letters, interdepartmental communications and reports. This requires the reporting component to access data from the files in the database component and to format it to meet document requirements. The interface of Lotus and WordPerfect templates is inherent in the design of the reporting component.

This conceptual overview of the prototype permit review system and its individual components shows the capabilities of the computer spreadsheet to go beyond the traditional usage of numerical computations. While the system's full design is only conceptual (meaning not all the components are not fully functional nor interfaces with each other) and is certain to change as the prototype is developed and implemented, the project does provide a "backdrop" for investigating the engineering uses and capabilities of computer spreadsheets.

Systems Analysis

In the above sections of this appendix, an introduction to surface water management and to a prototype permit review system was presented. These overviews were presented to show and define the problem that the prototype system must be able to solve. The complexity of the review process requires specific engineering knowledge along with managerial abilities to be employed by the reviewer.

As stated, the prototype permit review system has approached the permit review problem by breaking down and refining the process into workable modular components. The first usable component developed was the technical analysis portion. This consisted of several spreadsheet

templates that incorporated the modeling concepts found in already developed Basic programs. These programs allow the reviewer to readily check the design submitted on the application. These templates are good examples of Lotus's ability to be used as a capable replacement for traditional programing on a much more user-friendly basis which presents the user a complete picture of the calculation process without the "black box" effect. In addition, with the templates being tied together into the system's data bases, the design parameters and input will only have to be entered into the system once. Also, with the template's output transferable to the data base and with the future development of the checking component, the template's results will be able to be instantly compared and checked against the submitted application. These results and comparison would also be immediately available for direct insertion into correspondence and staff reports.

The process of using the data base's information for the generation of review documents and interdepartment communication is provided by the reporting component which consist of spreadsheet templates and WordPerfect files. The integration of the system's data base and word processing capabilities is again a good example of computer spreadsheet uses in the engineering field. It has been able to combine the two of the most powerful micro computer applications into a working system.

However, the key component of the system which has been most effective in demonstrating the full potential of computer spreadsheets to engineering is the implementation of the data base component. As the draft report on phase one of the surface permit review system has stated, the data base system is the "glue" that ties the parts together or the "back

bone" of the system which integrates and allows for the storage and transfer of data from a variety of sources. While the technical and reporting templates did not themselves represent any huge change to the current permit review procedures, the data base component is considered a significant development. By using @BASE data bases, the permit information is summarized in a form that is easily accessible for historical and locational review. Data supplied by the applicant can be stored in the data base for checking and analysis. The data bases records may also be used for other applications or permits which have similar characteristics. This will allow permit parameters and data allowed for or used on previous permits to become a more easily accessible. The data base will grow and become a more valuable data source as additional information can be taken from other manuals, textbooks, technical literature as well as from the inputted information of permit applications. At its most elementary level this will allow for consistency checks between parameter values submitted on current applications and those allowed in the past. Maybe, more significantly, the information submitted on permit applications through the review data base will provide a valuable data source for enforcement functions, planning studies, and policy decisions.

The above described material and more user information is well documented and explained in the prototype's phase one report.

That the prototype permit review system should offer a high pay back after it is fully implemented. By using the system the review engineer is no longer required to manually track permit applications or to continually re-enter input into BASIC programs and then hand copy the results into a

document on a word processor. Instead, all of these functions are integrated into a computer spreadsheet based system, and the information used or resulting from one functional element is easily transferred to another thus requiring less duplication of effort on the part of the engineer.

While the prototype system would not be able to review an application by itself, the aggregate of its capabilities does provide the review engineer with an automated stand alone system conveniently tied together by relational data bases. The system thereby allows the engineer to concentrate on the engineering and common sense aspects of the application versus the mundane clerical tasking which currently uses up the reviewing engineer's time.

Summary

As described herein, the prototype system uses spreadsheet analysis techniques which are combined with Lotus macros and the data base capabilities of dBase as a powerful method of organizing, analyzing, and documenting the permit review process for South Florida Surface Water Management District. This resulting application is considered to be a decision support system which greatly reduces the design calculation and clerical tasks required of the review engineer by the introduction of a multi-purpose relational data base system.

With the initial phase now completed, it is only a matter of refining and possible introduction of Expert System technique to obtain the project's stated goals.

REFERENCES

Gifford, J. Brent, "Microcomputers in Civil Engineering Use and Misuse", Journal of Computing in Civil Engineering, American Society of Civil Engineers, Vol. 1, No. 1 January, 1987.

Heaney, James P., Thomas Potter, and James Wittig, Draft Progress Report on the Surface Water Permit Review System (phase 1) Florida Water Resources Research Center, University of Florida, Gainesville, FL., September 1988.

If/Then Solutions, If/Then, Computer Software User's Manual, San Francisco, 1987.

Jorgensen, Carolyn, Mastering 1-2-3, SYBEX, San Francisco, 1988.

Maher, Mary Lou, Expert Systems for Civil Engineers: Technology and Application, American Society of Civil Engineers, New York, 1987.

Ortolano, Leonard and Catherine D. Perman, "Software For Expert Systems Development", Journal of Computing in Civil Engineering, American Society of Civil Engineers, Vol., 1 No. 4, New York, October 1987, pp. 224 - 234.

Partridge, D., Artificial Intelligence: Applications in the future of software engineering, John Wiley & Sons, Inc., New York, 1986.

Personics Corporation, @BASE Computer Software User's Manual, Concord, 1987.

Sell, Peter S., Expert Systems A Practical Introduction, John Wiley & Sons, Inc., New York, 1985.

Tootle, Glenn A., The Application of Computer Spreadsheets to Stormwater Management in the State of Florida, Master's Report Department of Civil Engineering, University of Florida, Spring 1987.

Wall, Richard A. Jr., Expert Systems in Civil Engineering, Master's Report, University of Florida, Summer 1986.

BIBLIOGRAPHY

Chu, L. Louis, Electronic Spreadsheet as an Engineer's Tool, Science Press, Beijing, China 1985.

Collier, Courtland A., and William B. Ledbetter, Engineering Economic and Cost Analysis, Second Edition, Harper & Row, Publishers, New York, 1988.

Gifford, J. Brent, "Microcomputers in Civil Engineering Use and Misuse", Journal of Computing in Civil Engineering, American Society of Civil Engineers, Vol. 1, No. 1 January, 1987.

Gilbert, Chris and Laurie Williams, The ABC's of 1-2-3, SYBEX San Francisco, 1986.

Godreau, E. Jessica, Methodologies for Problem Solving on Electronic Spreadsheets, Individual Study, Department of Environmental Energy, University of Florida, Gainesville, FL., 1987.

Hancock, Michael C., and James P. Heaney, "Water Resources Decisions Using Electronic Spreadsheets", Journal of Water Resource Planning and Management, Vol. 113, No. 5, September 1987, pp. 639 - 658.

Harmon, Paul and David King, Expert Systems Artificial Intelligence in Business, John Wiley & Sons, Inc., New York, 1985.

Heaney, J. P. Water Resources Engineering and Decision Making, Notes for ENV 4004, University of Florida, Gainesville, FL 1987.

Heaney, James P. and Leel Knowles, Microcomputer-Based Spreadsheet and CAD for Inventory and Analysis of Small Quantity Generators of Hazardous Waste, mimeograph report, Florida Water Resource Research Center, 1988.

Heaney, James P., Thomas Potter, and James Wittig , Draft Progress Report on the Surface Water Permit Review System (phase 1), Florida Water Resource Research Center, University of Florida, Gainesville, FL., September 1988.

If/Then Solutions, If/Then, Computer Software User's Manual, San Francisco, 1987.

Jorgensen, Carolyn, Mastering 1-2-3, SYBEX, San Francisco, 1988.

Maher, Mary Lou, Expert Systems for Civil Engineers: Technology and Application, American Society of Civil Engineers, New York, 1987.

Manheim, Marvin L., "A General Model for Providing Active Computer Support to Design, Planning, and Management", Microcomputers in Civil Engineering, Vol. 3, No. 1, March, 1988, pp. 29 - 40.

Mumpower, Jeryl L., Ortwin Renn, Lawerence D. Philips, and V.R.R. Uppuluri, Expert Judgement and Expert Systems, Springer-Verlag, New York, 1986.

Ortolano, Leonard and Catherine D. Perman, "Software For Expert Systems Development", Journal of Computing in Civil Engineering, American Society of Civil Engineers, Vol., 1 No. 4, New York, October 1987, pp. 224 - 234.

Partridge, D., Artificial Intelligence: Applications in the future of software engineering, John Wiley & Sons, Inc., New York, 1986.

Personics Corporation, @BASE Computer Software User's Manual, Concord, CA., 1987.

Resource Control Dept., South Florida Water Management District, Management and Storage of Surface Waters, Permit Information, Vol. IV, June 1987.

Scarlatos, Panagiotis D., "An Expert-System Approach to Hydrologic Data Fusion", Computing and Civil Engineering, American Society of Civil Engineering, 1987.

Sell, Peter S., Expert Systems A Practical Introduction, John Wiley & Sons, Inc., New York, 1985.

Tootle, Glenn A., The Application of Computer Spreadsheets to Stormwater Management in the State of Florida, Master's Report, Spring 1987.

Wall, Richard A. Jr., Expert Systems in Civil Engineering, Master's Report, University of Florida, Summer 1986.

Water Resources Research Group, If/Then Seminar Series, University of Florida, 1987.